162-TD-001-002

Science Software I&T Operational Procedures for the ECS Project (aka, The Green Book)

Technical Data

July 1997

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Stephen Berrick /s	7/11/97
Stephen Berrick, DAAC SSI&T Support	Date
EOSDIS Core System Project	

SUBMITTED BY

Karin Loya	7/11/97
Karin Loya, SSI&T Manager	Date
EOSDIS Core System Project	

Hughes Information Technology Systems
Upper Marlboro, Maryland

Abstract

The purpose of this Technical Data Paper (also referred to as the “Green Book”) is to delineate the operational procedures to accomplish the various steps that may be involved in the integration and test of Science Data Production Software (SDPS/W) with the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS). The SDPS/W integration and test (SSI&T) is performed at the Distributed Active Archive Center (DAAC) responsible for the generation of the standard products.

General information concerning preparing and delivering SDPS/W to the DAAC is found in the *Science User’s Guide and Operations Procedure Handbook for the ECS Project, Part 4: Software Developer’s Guide to Preparation, Delivery, Integration and Test with ECS* (205-CD-002-003). Each DAAC and Instrument Team (IT) combination have formulated specific agreements, understandings, or procedures that will guide their respective SSI&T activities. The procedures in this document provide detailed instructions on how to use the tools that are provided in the Pre-Release B Testbed of ECS to accomplish the steps outlined in the DAAC-IT procedures.

This page intentionally left blank.

Contents

12. PGE Planning, Processing, and Product Retrieval

13. File Comparison

14. Data Visualization

Tables

1. Introduction

1.1 Purpose

The purpose of this Technical Data Paper (also referred to as the “Green Book”) is to delineate the operational procedures to accomplish the various steps that may be involved in the integration and test of Science Data Production Software (SDPS/W) with the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS). The SDPS/W integration and test (SSI&T) is performed at the Distributed Active Archive Center (DAAC) responsible for the generation of the standard products.

General information concerning preparing and delivering SDPS/W to the DAAC is found in the *Science User’s Guide and Operations Procedure Handbook for the ECS Project, Part 4: Software Developer’s Guide to Preparation, Delivery, Integration and Test with ECS* (205-CD-002-003). Each DAAC and Instrument Team (IT) combination have formulated specific agreements, understandings, or procedures that will guide their respective SSI&T activities. The procedures in this document provide detailed instructions on how to use the tools that are provided in the Pre-Release B Testbed of ECS to accomplish the steps outlined in the DAAC-IT procedures.

1.2 Organization

Section 2 lists related documentation. Section 3 is devoted to general information concerning SSI&T including a road map for integrating a PGE into the ECS. Section 4 is deals with the Delivered Algorithm Package. Section 5 describes procedures for checking Earth Science Data Types (ESDTs) and generating new ones if necessary. Section 6 contains procedures related to configuration management. Section 7 describes set up, preparation, and use of the SSIT Manager, the main SSI&T tool provided at the DAACs. Section 8 begins procedures dealing with standards checking and compliance of the science software. Section 9 describes how to build the delivered science software and Section 10 describes how to run it in a simulated SCF environment.

Section 11 begins the procedures for integration of the science software into the ECS; it describes procedures for registering the PGE with the Planning and Data Processing System (PDPS) and for inserting data files to the IMF Data Server. Section 12 describes procedures for planning and running science software within the PDPS and retrieving the output. Section 13 then describes procedures for comparing the output produced to that delivered.

Section 14 contains procedures for examining output in more detail using some of the data visualization tools provided.

Section 15 describes procedures for activities that are post-SSI&T including problem reporting. Section 16 describes some procedures for investigating problems that occur during SSI&T either with the science software or with the ECS software.

Section 17 contains miscellaneous procedures that do not fit into the previous sections.

Appendix A describes the IMF Data Server in its handling of metadata. Appendix B contains a sample checklist file used with the SSIT Manager. Appendix C contains a table of environment variables that are needed and used by various SSI&T tools. Appendix D contains sample Target MCFs for a variety of files that are to be Inserted to the IMF Data Server. Finally, Appendix E contains sample ODL files that are used in the PGE registration process.

1.3 Review and Approval

This Technical Data Paper is an informal document approved at the Office Manager level. It does not require formal Government review or approval; however, it is submitted with the intent that review and comments will be forthcoming.

Questions regarding technical information contained within this paper should be addressed to the following ECS contact:

Karin Loya, SSI&T Manager

(301) 925-1052

kloya@eos.hitc.com

2. Related Documentation

The following related documentation is available either in hard copy from the author or generating source or on the World Wide Web (WWW) at the URL listed. Additionally, most documents generated by the ECS project are available on the WWW via the ECS Data Handling System at http://edhs1.gsfc.nasa.gov/.

2.1 Parent Documents

The parent documents are the documents from which this document's scope and content are derived.

162-TD-001-001	Science Software I&T Operational Procedures for the ECS Project (aka, the Green Book)
609-CD-001-001	Interim Release One (Ir1) Maintenance and Operations Procedures for the ECS Project
609-DR-002-001	Release A Operations Tools Manual for the ECS Project

2.2 Applicable Documents

The following documents are referenced within and are directly applicable to this document:

423-16-01	Data Production Software and Science Computing Facility (SCF) Standards and Guidelines, Revision A, October 1996
205-CD-002-003	Science User's Guide and Operations Procedure Handbook for the ECS Project, Volume 4: Software Developer's Guide to Preparation, Delivery, Integration and Test with ECS, Revision 1
333-CD-003-005	Release A SCF Toolkit Users Guide for the ECS Project
445-TP-006-001	EOSView Users Guide for the ECS Project
	FORCHECK for Sun/SunOS, A Fortran Verifier and Programming Aid, Installation Guide
	FORCHECK for Sun/SunOS, A Fortran Verifier and Programming Aid, Users Guide
	IDL Basics, v3.5, November 1993, Research Systems, Inc., Boulder, CO

IDL Reference Guide, v4, Volume 1, Routines A-M, March 1995, Research Systems, Inc., Boulder, CO

IDL Reference Guide, v4, Volume 2, Routines N-Z, March 1995, Research Systems, Inc., Boulder, CO

IDL User's Guide, v4, March 1995, Research Systems, Inc., Boulder, CO

PureDDTS (Distributed Defect Tracking System) 3.2.1 User's Manual, Pure Software (support@pure.com), Part Number PDT320-XPX-UGD

2.3 Information Documents

The following documents, although not referenced herein or directly applicable, do amplify or clarify the information presented in this document:

X3.9-1978	ANSI FORTRAN 77 Programming Language Standard
305-CD-010-001	Release A SDPS Planning Subsystem Design Specification for the ECS Project
416-TP-001-001	Implementation Plan for the Pre-Release B Testbed for the ECS Project
163-WP-001-002	An ECS Data Provider's Guide to Metadata

Please note that Internet links cannot be guaranteed for accuracy or currency.

World Wide Web pages…

Prohibited Functions in the EOSDIS Core System at URL: http://ecsinfo.hitc.com/iteams/ProhibFunc/

FORTRAN 77 in the EOSDIS Core System at URL: http://ecsinfo.hitc.com/iteams/Standards/F77std.html

PGE Design Information Page at URL: http://ecsinfo.hitc.com/iteams/Science/pge_wp.html

EOS Instrument Team Information Page at URL: http://ecsinfo.hitc.com/iteams/

3. General Information

3.1 SSI&T Road Map

Science software integration and test (SSI&T) is the process by which science software is tested for production readiness in the DAACs. The goals are to assure (1) *reliability*, which means that the software runs to normal completion repeatedly over the normal range of data inputs and runtime conditions, and (2) *safety*, which means that the software executes without interfering with other software or operations.

SSI&T activities can be broadly separated into two categories: pre-SSI&T and formal SSI&T. Pre-SSI&T activities can be defined as those activities that do not involve the ECS Planning and Data Processing System (PDPS) or the IMF Science Data Server. Formal SSI&T activities are defined as those that involve the full ECS including the PDPS and IMF Science Data Server.

Table 3.1-1 can be used as a PGE road map for the remainder of the Green Book (a "30 Steps To a Healthier, Happier PGE" program). It lists most SSI&T activities in the order in which they are most likely to occur. Column 1 describes the SSI&T task; column 2 lists the relevant sections in the Green Book; and column 3 represents an attempt to qualify the importance of the task. Tasks labeled as "Critical" must be done in order to have the PGE integrated into the ECS and run by the automated PDPS. Tasks labeled as "Essential", "Valuable", and "Optional" have lowering degrees of importance, respectively, and are somewhat subjective. This table should not be misconstrued as overriding specific agreements between DAACs and Instrument Teams.

Table 3.1-1 Road Map of SSI&T for a PGE

Step	Task	Sections	Importance
1	Acquire and unpack the Delivered Algorithm Package (DAP).	4	Critical
2	Inspect the delivery and accompanying documentation.	N/A	Valuable
3	Place the DAP contents under configuration management.	6	Valuable
4	Check the source files for standards compliance.	8.1,8.2, 8.3, 8.4	Valuable
5	Check the source files for prohibited functions.	8.5	Valuable
6	Extract and check prologs.	8.7	Valuable
7	Check the Process Control File (PCF).	8.6	Valuable

Table 3.1-1 Road Map of SSI&T for a PGE (cont.)

Step	Task	Sections	Importance
8	Revise the delivered PCF to comply with the local DAAC environment and the location of data files.	9.1	Essential
9	Compile the SDP Toolkit Status Message Facility (SMF) files.	9.3	Essential
10	Build the PGE linking it to the SCF version of the SDP Toolkit.	9.4	Essential
11	Run and profile the PGE from the command line. Save the profiling results. They will be used later when entering operational metadata into the PDPS.	10.1, 10.2	Essential
12	Check the exit status of the PGE and examine the PGE log files for errors or anomalous messages.	16.1.1	Essential
13	Compare the output produced with test data delivered with the PGE.	13.1, 13.2, 13.3, 13.4	Essential
14	Examine the output products using data visualization techniques.	14	Optional
15	Verify that delivered MCFs (Source MCFs) have content compatible with the granule level metadata sections of the corresponding ESDT descriptor files. If not compatible, update the descriptor files.	5.1.1, 5.1.2	Critical
16	Verify that all required ESDTs have either: (1) Been successfully registered within the ECS, or (2) Can be registered with existing descriptor files If an ESDT has not been registered, then register it. If a descriptor file does not exist, then make one and register it. ESDTs are needed for: (1) All input and output data granules (2) All MCFs (one ESDT can be used for all) (3) The PGE executable tar file (SSEP)	5.2, 5.3	Essential
17	For each ESDT used by the PGE, construct an ESDT ODL file for updating the PDPS or verify that they already exist. ESDT ODL files are needed for:	11.1.1	Critical

Table 3.1-1 Road Map of SSI&T for a PGE (cont.)

Step	Task	Sections	Importance
	(1) All input and output data granules (2) All MCFs (3) The PGE executable tar file (SSEP) This begins PGE registration.		
18	Construct a PGE ODL file for updating the PDPS database. This involves using the updated PCF to construct an initial PGE ODL file. The PGE ODL file must then be hand edited to add required metadata. A mapping between logical IDs in the PCF and ESDT ShortNames must be known *before* this step is done.	11.1.2	Critical
19	Supply operational metadata to the PDPS database. This completes the PGE registration.	11.1.3	Critical
20	Build the PGE linking it to the DAAC version of the SDP Toolkit.	9.5	Critical
21	For each input dynamic data granule needed by the PGE, construct a Target MCF and Insert it to the IMF Data Server.	11.2.1, 11.2.2	Critical
22	For each input static data granule, construct a Target MCF and Insert it to the IMF Data Server. Such a Target MCF is needed in order to Insert: (1) All input static data granules (2) All MCFs (3) The PGE executable tar file (SSEP)	11.2.3, 11.2.4	Critical
23	Assemble the SSEP (as a tar file) and Insert it to the IMF Data Server.	11.3.1, 11.3.2	Critical
24	Initiate a Production Request (PR) that will result in a single Data Processing Request (DPR), that is, run once.	12.2	Critical
25	Register a subscription for the PGE input and output files.	12.1	Critical
26	Use the Planning Workbench to plan the PR and hence, run the PGE.	12.3	Essential
27	Monitor the PGE's progress using AutoSys.	12.4	Essential

Table 3.1-1 Road Map of SSI&T for a PGE (cont.)

Step	Task	Sections	Importance
28	Examine the Production History (PH) including the PGE log files for errors or anomalous messages.	16.1.2, 16.2	Essential
29	Compare the output produced with test data delivered with the PGE.	13.1, 13.2, 13.3, 13.4	Essential
30	Examine the output products using data visualization techniques.	14	Optional

3.2 Set Up of SSI&T User Accounts

During EGS testing at the Goddard DAAC and early SSI&T of MODIS PGEs, much has been learned about the setting up SSI&T user accounts on the Pre-Release B Testbed. This knowledge has been used to informally develop a number of set up scripts which allow SSI&T users to have the proper environments on all machines accessed during SSI&T.

These set up scripts are under informal CM (using RCS) and are available as a compressed UNIX tar file at:

ftp://ecsinfo.hitc.com/Testbed/ssit_account_setup.tar.Z

A README file is included in the tar archive.

These files are available as informational only. There is no assumption in these Green Book procedures that they are being used.

3.3 How to Use The Procedures

The science software I&T operational procedures contained within this document are ordered. The order is intended to *loosely* suggest a logical sequence which, when used as a "road map", represents an overall, sensible end-to-end SSI&T activity at the Pre-Release B Testbed DAACs. The ordering cannot, however, be interpreted as a detailed, step-by-step guide to SSI&T activities. In addition, since there are many factors that affect the actual SSI&T activities (*e.g.* Instrument Team deliveries, DAAC policies, agreements between the Instrument Teams and the DAACs), the ordering in this document can only be suggestive.

Each procedure document is broken up into three sections. The first section contains the Activity Checklist. This is a table containing, in broad terms, the steps necessary for carrying out the procedure. It is intended to give an overview.

The second section describes each of these steps in more detail. This section is geared to someone unfamiliar with the procedure or someone needing verbose descriptions. Conventions used in this section are as follows: Text shown in **bold** represents what is to be typed in literally via the keyboard. Text shown in ***bold italics*** represents something to be filled in by something appropriate. In all cases, the carriage return or “enter” key is represented as **Return**. For example:

cd ***MyDir***, press **Return**.

means type “cd” followed by a directory name which is appropriate to the task, and followed by pressing the return key.

Text shown in **bold Helvetica** font represents names, labels, buttons, etc. on graphical user interfaces (GUIs). For example:

In the GUI labeled **Process Control File Checker**, click on the **OK** button.

The third section contains the Quick Step Procedures. This table is geared to someone already familiar with the procedure and shows a condensed view of the steps required. Experienced users can jump directly to this section to quickly be reminded of the steps.

This page intentionally left blank.

4. Acquiring the Delivered Algorithm Package

Science software is delivered to SSI&T at the DAACs in the form of one or more Delivered Algorithm Packages (DAPs). The DAP contains science software source code, build or make files, documentation, and test data to test the software once built. Typically, a DAP corresponds to one PGE. The details as to the form and contents of a DAP should be according to documented DAAC and Instrument Team agreements.

The delivery mechanism for DAPs can be electronic (e.g. via UNIX ftp) or physical media such as 4mm or 8mm magnetic tape.

The procedures that following describe how to acquire a DAP and get it into a form for which SSI&T can begin.

4.1 Acquiring a DAP via FTP Pull

A DAP may be acquired electronically using the UNIX *ftp* (file transfer protocol) utility. This will typically only be used in cases where the size of the DAP is relatively small, less than several tens of megabytes (MB). Larger DAPs would take a prohibitively long time to transfer, in which case delivery via physical media may be more efficient.

An ftp pull activity is one that is initiated by the receiver, in this case, an SSI&T personnel member at the DAAC. An ftp push activity is one that is initiated by the sender, in this case, an Instrument Team member at the SCF. This procedure only addresses an ftp pull activity.

The Activity Checklist table that follows provides an overview of the process for acquiring a DAP. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 4.1-1 Acquiring a DAP via FTP Pull - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the delivery directory.	(I) 4.1	
2	SSI&T	Make an ftp connection to the remote SCF machine.	(I) 4.1	
3	SSI&T	Enter user login name.	(I) 4.1	
4	SSI&T	Enter password.	(I) 4.1	

Table 4.1-1 Acquiring a DAP via FTP Pull - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
5	SSI&T	Change directories to the location of the DAP(s)	(I) 4.1	
6	SSI&T	Change to binary mode transfer.	(I) 4.1	
7	SSI&T	Retrieve DAP.	(I) 4.1	
8	SSI&T	Close the ftp connection.	(I) 4.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. A directory at the DAAC has been set aside to receive DAPs via electronic transfer.
2. The SCF has notified the DAAC that DAPs are ready to be transferred from an agreed-upon machine and directory.
3. The procedures make no assumption as to the form or contents of the DAP nor do they address the process for SCF notification and DAAC acceptance.

To retrieve a DAP via ftp pull, execute the procedure steps that follow:

1 At a UNIX prompt, type **cd *DeliveryPathname***, press **Return**
- The ***DeliveryPathname*** is the full path name to the directory that has been set aside for ftp pull of DAPs from the Instrument Team. For example, **cd /inbox/ASTER**, press **Return**.

2 At a UNIX prompt, type **ftp *machineIPaddress***, press **Return**.
- The ***machineIPaddress*** is the IP address or fully qualified domain name of the remote SCF machine. For example, **ftp 192.116.53.2**, press **Return**.
- The remote machine will likely display some messages and then prompt for a login name.

3 At the ftp prompt on the remote machine, enter user login name, press **Return**.
- The ftp prompt will look like **Name (*machinename*:*username*):** where ***machinename*** will be the name of the remote machine and ***username*** will be the user name of the user initiating the ftp connection.
- Enter a user login name which should be known.
- The remote machine will typically respond with **331 Password required for *username*:**

4 At the ftp prompt on the remote machine, enter user password, press **Return**.
- The ftp prompt will look like **Password:**
- Enter the password which should be known.

- The remote machine will typically respond with **230 User *username* logged in** and display the **ftp>** prompt for further ftp commands.

5 At the ftp prompt on the remote machine, type **cd *DAPpathname***, press **Return**.
- The ***DAPpathname*** is the full path name to the directory on the remote machine containing the DAP to retrieve. For example, **cd /pub/outbox/**, press **Return**.
- The directory location should be known.

6 At the ftp prompt on the remote machine, type **binary**, press **Return**.
- The **binary** command causes subsequent file transfers to be in binary mode, preserving the integrity of the file to retrieve without interpretation (as would be done in ASCII mode.
- The system will typically respond with the message **200 Type set to I** indicating that binary mode has been set.

7 At the ftp prompt on the remote machine, type **get *DAPfilename***, press **Return**.
- The ***DAPfilename*** is the file name of the DAP to retrieve. For example, type **get ASTER_PGE3_V2.0_12151998.tar.Z**, press **Return**.
- The user may need to type **dir** and then **Return** to display a listing of the files in the current directory.
- The system will likely display several lines of messages once the transfer has completed. For large files, this may take a long time (minutes to hours depending upon the size of the DAP and the bandwidth of the connection).

8 At the ftp prompt on the remote machine, repeat step 7 or type **quit**, press **Return**.
- Typing **quit** and pressing **Return** closes the ftp connection with the remote machine.
- Retrieve other DAP files by repeating step 7.
- The DAPs retrieved will reside in ***DeliveryPathname*** on the local machine.

Table 4.1-2 Acquiring a DAP via FTP Pull - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd DeliveryPathname	press Return
2	ftp machineIPaddress	press Return
3	user login name	press Return
4	password	press Return
5	cd DAPpathname	press Return
6	binary	press Return
7	get DAPfilename	press Return
8	quit	press Return

4.2 Unpacking a DAP

Once a DAP has been acquired via electronic means or physical media, it typically needs to be unpacked before its contents are accessible for SSI&T. Several mechanisms are available under standard UNIX for packing and unpacking files to and from a file archive, the most common being UNIX *tar*. Another fairly typical utility is *gzip* and its companion, *gunzip*.

The file name extension is usually an indication of the packing utility used and DAP files should use this convention. DAP files that have been packed using the UNIX *tar* utility will usually have .tar as a file name extension indicating a tar file. If the DAP has been further compressed using the UNIX *compress* utility, the file name extension is typically .tar.Z indicating a compressed tar file. For DAP files packed with the *gzip* utility, the .zip file name extension is generally used.

When unpacking is performed on a DAP, the contents of the packed file are moved from the tar archive to local disk. If the DAP tar file contains directories as well as files, these directories will be created in the same structure as in the tar file. This structure typically reflects the directory structure from which the tar file was created in the first place at the SCF. Once a tar file has been unpacked, the original tar file will still exist unaltered.

In order to unpack a DAP, the utility for doing so must be known. The file name extension may not always indicate how a DAP was packed or even if it was packed at all. This procedure describes how to unpack DAPs that have been packed with the UNIX *tar* utility only, with or without additional compression.

The Activity Checklist table that follows provides an overview of the process for unpacking a DAP. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 4.2-1 Unpacking a DAP - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the location of the packed DAP file.	(I) 4.2	
2	SSI&T	If necessary, uncompress the DAP file.	(I) 4.2	
3	SSI&T	Unpack the DAP file.	(I) 4.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. A directory has been set aside for unpacking the DAP file.

To unpack a DAP, execute the procedure steps that follow:

1 At a UNIX prompt, type **cd *UnpackPathname***, press **Return**

- The ***UnpackPathname*** is the path name of the directory that has been set aside for unpacking of DAPs. For example, **cd /staging/ASTER**, press **Return**.

2 If necessary, at a UNIX prompt, type **uncompress *PackedDAP.Z***, press **Return**.

- The ***PackedDAP.Z*** is the file name of the compressed DAP file. For example, **uncompress ASTER_PGE3_V2.0_12151998.tar.Z**, press **Return**.
- The file name extension of **.Z** is a convention indicating UNIX compressed files. The **uncompress** utility expect this file name extension by default.
- A resulting error may indicate that the DAP file was not compressed or that another compression utility was used.
- If the file name extension was **.Z**, the uncompressed version will have the same file name but without the **.Z**, for example ***PackedDAP***.

3 At the UNIX prompt, type **tar xvf *PackedDAP***, press **Return**.

- The ***PackedDAP*** is the file name of the uncompressed DAP file. For example, **tar xvf ASTER_PGE3_V2.0_12151998.tar**, press **Return**.
- The tar archive will be unpacked in the current directory. If the archive contained directories and subdirectories, these will be created by the tar utility and populated by the files that belong.

Table 4.2-2 Unpacking a DAP - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd UnpackPathname	press Return
2	(If necessary) uncompress PackedDAP.Z	press Return
3	tar xvf PackedDAP	press Return

This page intentionally left blank.

5. Preparation of Earth Science Data Types

Every science data product generated and archived by the ECS must be described to the system by metadata which are put into an inventory and then used to retrieve and distribute the data to users of the system. The ECS Earth Science Data Model organizes the metadata into groups of related attributes and services to be performed on the data products. Granules of the same type of science data are grouped into collections. Every collection is described by an Earth Science Data Type (ESDT) and is made known to the system by an ESDT descriptor file and associated software code which is built into the Data Server's dynamic link library (DLL) to perform the services. The ESDT descriptor is composed of sections containing the following information:

- Collection level metadata attributes with values contained in the descriptor.
- Granule level metadata attributes whose values are supplied primarily by the Product Generation Executives (PGEs) during runtime.
- Valid values and ranges for the attributes.
- List of services for the data and events which trigger responses throughout the system.

The ESDTs for all data collections to be input to or output from the PGEs must be built and registered into the ECS before any of the PGEs can be run under the automated processing system. Since the original Release A Data Server is no longer a part of the Pre-Release B Testbed, the DLL code is no longer required.

During the past year, the ECS has collected information from the Instrument Teams on the ESDTs needed for their science software in the Pre-Release B Testbed. This information has been baselined and a set of ESDT descriptor files have been built and tested according to this baseline. The baselined ESDT descriptor files are assumed to reside in the Testbed under CM using ClearCase. Since science software may have evolved during the time between the ESDT baseline and delivery to the DAAC, some changes to the baselined ESDT descriptor files should be expected. In addition, some new ESDTs may be required which were not included in the ESDT baseline.

A set of SSI&T ESDT Tools have been developed to automate the process of making changes to existing ESDTs. The procedures and tools that follow describe how to assess whether changes have been made relative to the baselined ESDTs and if so, how to make new versions of these ESDTs. Another tool allows the generation of new ESDTs from scratch.

5.1 Comparing Granule Level Metadata

A PGE accesses granule level metadata attributes and values via a Source Metadata Configuration File (MCF). There is typically one Source MCF for each output data set. The ESDTs which have been built and registered in the ECS contain a section for granule level metadata attributes and values for each data set. In terms of content, the MCFs and the granule level metadata section of the corresponding ESDT descriptor files have to be in sync. As expressed in the introductory paragraphs (above), the science software in the PGEs may have changed to a point where these two locations of granule level metadata are out of sync.

Few changes are expected in the Inventory section of the Source MCF. Changes are more likely to be expected in the Archive section of the Source MCF and in the Product Specific Metadata. If there are *any* changes, a new version of the baselined ESDT descriptor file must be generated and should be placed under CM.

Section 5.1.1 describes how to determine if the delivered Source MCF and the registered ESDT descriptor file are in sync with one another. Section 5.1.2 describes how to generate a new version of an existing ESDT descriptor file if the delivered Source MCF is found not to be in sync with the ESDT descriptor file.

5.1.1 Comparing Granule Level Metadata in ESDT to Delivered Source MCF

The MCFToDescChecker tool compares a delivered Source MCF with the granule level metadata section of an ESDT descriptor file. The program requires as input the delivered Source MCF and a copy of the existing ESDT descriptor from CM. Execution of this program will indicate whether the granule level metadata content is consistent between the Source MCF and the ESDT descriptor file or it will produce diagnostics indicating where differences have been detected.

The Activity Checklist table that follows provides an overview of the process for comparing a delivered Source MCF to the granule level metadata in the ESDT descriptor file. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 5.1.1-1 Comparing Granule Level Metadata in ESDT to Delivered Source MCF - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view to the ESDT under CM.	(I) 6.2	
2	SSI&T	Run the MCFToDescChecker tool.	(I) 5.1.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Source MCF is available and accessible.
2. There exists an ESDT descriptor file under CM in ClearCase corresponding to this Source MCF.
3. The VA has granted necessary ClearCase privileges for the user.

To compare the granule level metadata in the ESDT to the delivered Source MCF, execute the procedure steps that follow:

1 At a UNIX prompt on an AIT Sun, type **cleartool setview *ViewName***, press **Return**.
- The ***ViewName*** is the name of the view that allows access to the ESDT descriptor file under CM.
- For example, type **cleartool setview esdtvob**, press **Return**.

2 At a UNIX prompt on the AIT Sun, type **MCFToDescChecker *pathname*/*ESDTfilename MCFfilename,*** press **Return**.
- The ***pathname*** is the full path name to the location of the ESDT descriptor file (in the VOB).
- The ***ESDTfilename*** is the file name of the ESDT descriptor file. By convention, the file name is the ESDT ShortName.
- The ***MCFfilename*** is the file name of the delivered Source Metadata Configuration File (MCF) with which to compare the ESDT descriptor file.
- For example, type **MCFToDescChecker /DAAC/ESDT/CER/CER01aT CER01aT.MCF**
- After the program has run, a `Identical` will be displayed if the Source MCF and ESDT descriptor file are in sync or diagnostics will be displayed if the two files are not in sync.

Table 5.1.1-2 Comparing Granule Level Metadata in ESDT to Delivered Source MCF - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool setview ViewName	press Return
2	MCFToDescChecker pathname/ESDTfilename MCFfilename	press Return

5.1.2 Generating a New Version of ESDT from Delivered Source MCF

If the delivered Source MCF is found to be incompatible with the baselined ESDT descriptor file, the UpdateDesc tool can be used to update the original ESDT descriptor file to a new version. This new version will have the same ShortName, but a new VersionID. The UpdateDesc tool requires as input the ESDT descriptor file and the delivered Source MCF. The new ESDT version will be constructed with a granule level metadata section that is compatible with the delivered Source MCF. To verify that the new ESDT descriptor file has been correctly updated, the MCFToDescChecker tool (see Section 5.1.1) can be run again using the new ESDT descriptor file.

The Activity Checklist table that follows provides an overview of the process for creating a new version of the ESDT descriptor file based on the delivered Source MCF. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 5.1.2-1 Generating a New Version of ESDT from Delivered Source MCF - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view to the ESDT VOB.	(I) 6.2	
2	SSI&T	Check out the ESDT descriptor file.	(I) 5.1.2	
3	SSI&T	Run the UpdateDesc tool.	(I) 5.1.2	
4	SSI&T	Verify the update using the MCFToDescChecker tool.	(I) 5.1.2	
5	SSI&T	Rename the modified ESDT descriptor file.	(I) 5.1.2	
6	SSI&T	Check in the modified ESDT descriptor file.	(I) 5.1.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The UpdateDesc tool has been installed on the AIT Sun and is available.

2. The Source MCF is available and accessible.
3. There exists an ESDT descriptor file under CM in ClearCase corresponding to this Source MCF.
4. The VA has granted necessary ClearCase privileges for the user.

To generate a new version of an ESDT from the delivered Source MCF, execute the procedure steps that follow:

1 At a UNIX prompt on an AIT Sun, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of the view that allows access to the ESDT under CM.
- For example, type **cleartool setview esdtvob**, press **Return**.

2 At a UNIX prompt on the AIT Sun, type **cleartool checkout -nc *pathname*/*ESDTfilename***, press **Return**.

- The ***pathname*** is the full path name to the location of the ESDT descriptor files.
- The ***ESDTfilename*** is the file name of the ESDT descriptor file that will be modified.
- The **-nc** stands for “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check out step.
- For example, type **cleartool checkout -nc /DAAC/ESDT/CER/CER01aT**

3 At a UNIX prompt on the AIT Sun, type **UpdateDesc *ESDTfilename MCFfilename ESDTfilename*.new,** press **Return**.

- The ***ESDTfilename*** is the file name of the ESDT descriptor file to be modified.
- The ***MCFfilename*** is the file name of the delivered Metadata Configuration File (Source MCF).
- The ***ESDTfilename*.new** is the file name to be given to the output ESDT descriptor file. The file name of the ESDT descriptor file, by convention, is the ESDT ShortName. The file name extension .new indicates that it is a new version.
- For example, type **UpdateDesc CER01aT CER01aT.MCF CER01aT.new**, press **Return.**

4 At a UNIX prompt on the AIT Sun, type **MCFToDescChecker *ESDTfilename*.new *MCFfilename***, press **Return**.

- The ***ESDTfilename*.new** is the file name of the modified ESDT descriptor file created in step 5.
- The ***MCFfilename*** is the file name of the delivered Metadata Configuration File (Source MCF).
- This step verifies that the new ESDT descriptor file has identical granule level metadata content identical to the delivered MCF and that the Update Descriptor File Tool was successful.
- For example, type **MCFToDescChecker CER01aT.new CER01aT.MCF**, press **Return.**

5 If the **MCFToDescChecker** returned a `Identical`, at a UNIX prompt on the AIT Sun, type **mv *ESDTfilename*.new *ESDTfilename***, press **Return**.

- The ***ESDTfilename*.new** is the file name of the modified ESDT descriptor file created in step 4.
- The ***ESDTfilename*** is the file name of the original ESDT descriptor file and the file name to be given to the modified version.
- This command renames the modified version of the ESDT descriptor file to the original file name.
- For example, type **mv CER01aT.new CER01aT**, press **Return**.
- Depending upon the user environment, the shell may ask for confirmation since the original version of the descriptor file will be overwritten.

6 At a UNIX prompt on the AIT Sun, type **cleartool checkin -c "*comment*" *pathname*/*ESDTfilename*,** press **Return.**

- The ***pathname*** is the full path name to the location of the ESDT descriptor files in the ClearCase VOB. The directory must be checked out before the file to modify.
- The ***ESDTfilename*** is the name of the now modified ESDT descriptor file.
- The **-c "*comment*"** indicates a comment to be associated with the check in. The comment must be enclosed within double quotes and immediately follow the **-c** flag.
- This command checks in the modified ESDT descriptor file to ClearCase.
- For example type, **cleartool checkin -c "Modified according to delivered MCF at Pre-Release B Testbed" /DAAC/ESDT/CER/CER01aT,** press **Return**. Alternatively, use the **-nc** flag to enter in no comment.

Table 5.1.2-2 Generating a New Version of ESDT from Delivered Source MCF - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool setview ViewName	press Return
2	cleartool checkout -nc /pathname/ESDTfilename	press Return
3	UpdateDesc ESDTfilename MCFfilename ESDTfilename.new	press Return
4	MCFToDescChecker ESDTfilename.new MCFfilename	press Return
5	mv ESDTfilename.new ESDTfilename	press Return
6	cleartool checkin -c "comment" pathname/ESDTfilename	press Return

5.2 Creating a New ESDT

In some cases, PGEs will require ESDTs that are not part of the ESDT baseline and are therefore not available at the DAAC. In these cases, new ESDTs will have to be built from scratch. The tools provided for generating new ESDTs assume limited collection level metadata (six attributes). These tools can generate ESDTs for dynamic data granules (those that are associated with a temporal locality) or static data granules (typically, permanent files like lookup tables or coefficient files).

Generation of a new ESDT requires several steps. First, a comma-delimited text file containing the six collection level attribute values is needed. Such a file is most easily generated using a spreadsheet program like MS Excel. This procedure is described in Section 5.2.1.

Second, a Source MCF corresponding to each new ESDT is required. For dynamic granules, at least, Source MCFs should be part of the delivery to the DAAC. If the data granule is static, however, then it is unlikely that a corresponding MCF has been delivered. In this case, a default template MCF will be assumed. Note that Source MCFs are *not* checked for correctness by this tool and are assumed to be correct. The procedures for using this tool are described in Section 5.2.2.

If ESDTs require Product Specific Attributes (PSAs), these will have to be added to the generated ESDT descriptor file manually after it is been constructed.

Every file that is to be Inserted to or Acquired from the IMF Data Server must have some ESDT defined for it.

5.2.1 Generating a Comma-Delimited Text File from a Spreadsheet

In order to generate a new ESDT descriptor file, a comma-delimited text file must be created. Such a file can be created from a MS Excel spreadsheet.

The Activity Checklist table that follows provides an overview of the process for creating a comma-delimited text file from a spreadsheet. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 5.2.1-1 Generating a Comma-Delimited Text File from a Spreadsheet - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the MS Windows emulator.	(I) 5.2.1	
2	SSI&T	Invoke MS Excel from MS Windows.	(I) 5.2.1	
3	SSI&T	Create column headers in spreadsheet.	(I) 5.2.1	
4	SSI&T	Populate spreadsheet with collection level metadata.	(I) 5.2.1	
5	SSI&T	Invoke the Save dialog box.	(I) 5.2.1	
6	SSI&T	Save the spreadsheet as a comma-delimited text file.	(I) 5.2.1	
7	SSI&T	Quit MS Excel and MS Windows.	(I) 5.2.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager is running. This is required to bring up the MS Windows emulator and then MS Excel.

To generate a comma-delimited text file from a spreadsheet, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Office Automation** and then **MSWindows**.

- The MS Windows emulator GUI will be displayed.

2 Within the MS Windows GUI, double click on the MS Excel icon.

- The MS Excel program will start and a blank spreadsheet will be opened.

3 In the MS Excel GUI, enter in the following column headings into cells A1, B1, C1, D1, E1, F1, and G1: `ShortName`, `LongName`, `VersionID`, `CollectionDescription`, `ProcessingCenter`, `ArchiveCenter`, `Permanent`

- Enter the column headings as shown, one per cell in the default font.
- The tab key may be used to move the cursor from one cell to the next cell to the right.

4 Beginning in row B, populate the spreadsheet with values corresponding to the column headings entered above. These column headings refer to collection level metadata (a minimum set) that are necessary to later create an ESDT descriptor file. Each row entered will correspond to one ESDT to be created.

- `ShortName` must be a string less than or equal to eight characters.
- `LongName` must be a string less than or equal to 80 characters.
- `VersionID` must be a string less than or equal to 20 characters.
- `CollectionDescription` must be a string less than or equal to 255 characters.
- `ProcessingCenter` must be set to one of: `EDC`, `GSFC`, `LaRC`, or `NSIDC`.
- `ArchiveCenter` must be set to one of: `EDC`, `GSFC`, `LaRC`, or `NSIDC`
- `Permanent` must be set to one of: `Yes` or `No`. A `Yes` indicates that a MCF is **not** available for making an ESDT for this entry (*e.g.* because the file is static); a `No` indicates that a MCF **is** available for making an ESDT for this entry (the MCF will be used later).
- Information for populating these fields should be provided by the Instrument Teams.
- Do **not** insert comments or superfluous information between the column headings and the data or within the data area.

5 Once the spreadsheet has been populated with collection level metadata information for each ESDT to generate, click on the **File** menu, then choose **Save As…**.

- This will bring up a dialog box labeled **Save As**.

6 In the **Save As** dialog box, enter a file name in the field next to **File name:** under which to save the comma-delimited text file. Then in the **Save as type:** menu, select **CSV (comma delimited)(*.csv)** item. Finally, click on the **Save** button.

- Optionally, the field next to **Save in** may be used to specify a directory in which to save the file.
- The comma-delimited text file version of the spreadsheet created will be saved to local disk.

7 In the MS Excel GUI, click on the **File** menu, then choose **Exit**.

- This will close MS Excel.
- Optionally, quit MS Windows by clicking on the **File** menu in the Program Manager GUI, then choose **Exit**. A confirmation dialog box will be displayed; choosing **OK** will end the session with MS Windows.

Table 5.2.1-2 Generating a Comma-Delimited Text File from a Spreadsheet - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Office Automation → MSWindows	(No action)
2	MS Excel icon	double click
3	Headers: *ShortName*, *LongName*, *VersionID*, *CollectionDescription*, *ProcessingCenter*, *ArchiveCenter*, *Permanent flag*	(No action)
4	Enter collection level metadata values	(No action)
5	File → Save As…	(No action)
6	*filename*	save as type: comma delimited (CSV)
7	File → Exit	(No action)

5.2.2 Generating a New ESDT Descriptor File

The DescGen tool provides the capability to generate new ESDT descriptor files with limited collection level metadata only. This program is written in the C language and is designed to be run from the command line to produce one or more new ESDT descriptor files depending on the number of rows of *ShortName* for the collection and other attributes supplied in the metadata text file (see Section 5.2.1). The new ESDT descriptor files should then be placed under CM in ClearCase. DescGen can be packaged into a script with the ClearCase commands for a completely automated procedure for creating and configuring multiple new descriptor files. DescGen requires several input files containing metadata and several templates for the ESDT descriptor syntax and format required to produce the output ESDT descriptor files.

The following list contains the input files required for DescGen.

- The text file containing limited collection level metadata created in Section 5.2.1. DescGen reads the attribute values from this comma delimited text file and inserts them into the collection metadata section of the ESDT descriptor template. The value in the last column indicates whether the ESDT descriptor is to be generated for a static or permanent file requiring no MCF for the program input or a dynamic data granule for which an MCF must be provided as program input.
- The MCFs which are delivered with the PGE for new output products. The MCFs contain the granule level attributes which DescGen will insert into the granule level metadata section of the ESDT descriptor template. The program assumes that there is a MCF with the file name *ShortName*.MCF for each ESDT to be generated.

The following list contains the input templates required for DescGen.

- TemplateColl.odl is an ESDT descriptor template file defining limited collection level metadata attributes for dynamic data granule type of files for which an MCF with the granule level attributes will be provided as input to the program.
- TemplateCollYes.odl is an ESDT descriptor template file for limited collection level metadata attributes for static or permanent type of files for which no MCF will be provided as input to the program.
- TemplateService.odl is a template for the Service Group in the ESDT descriptor file. It will contain only generic services provided by ECS at the Pre-Release B Testbed.

The following paragraph describes the output file from DescGen.

- DescGen will generate one output ESDT descriptor file for each row in the collection level metadata text file. By convention at the Testbed, the ESDT descriptor file name is *ShortName.*

The Activity Checklist table that follows provides an overview of the process for generating a new ESDT descriptor file. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 5.2.2-1 Generating a New ESDT Descriptor File - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Run the DescGen tool on the collection level metadata text file.	(I) 5.2.2	
2	SSI&T	Use text editor to verify that the ESDT descriptor is correct.	(I) 5.2.2	
3	SSI&T	Set the ClearCase view to the ESDT VOB.	(I) 5.2.2	
4	SSI&T	Change directories to the location of the ESDTs relevant to the instrument.	(I) 5.2.2	
5	SSI&T	Copy over the new ESDT descriptor file to be checked into ClearCase.	(I) 5.2.2	
6	SSI&T	Check out the current directory from ClearCase.	(I) 5.2.2	
7	SSI&T	Create a new ClearCase element for the new ESDT descriptor file.	(I) 5.2.2	
8	SSI&T	Check in the new ESDT descriptor file.	(I) 5.2.2	
9	SSI&T	Check back in the current directory.	(I) 5.2.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The DescGen tool has been installed on the AIT Sun and is available.
2. A comma delimited text file has been produced according to procedure 5.2.1 and is accessible.
3. If product specific metadata are needed, these attributes will have to be entered manually.
4. ESDT descriptor files exist under CM in ClearCase.
5. The VA has granted necessary ClearCase privileges for the user.

To generate an ESDT descriptor file, execute the procedure steps that follow:

1 At a UNIX prompt on an AIT Sun, type **DescGen *TextFilename***, press **Return.**
- The ***TextFilename*** is the file name of the collection level metadata text file in comma-delimited format created in Section 5.2.1.
- For example, type **DescGen CERESDT.csv**, press **Return**.

2 At a UNIX prompt on the AIT Sun, type **vi *DescFilename***, press **Return.**
- The ***DescFilename*** is the file name of the ESDT descriptor file created in step 1. The file name will be the ESDT's ShortName.
- For example, type **vi CER01aT**, press **Return**.

- Any text editor may be used such as *emacs*. For example, **emacs CER01aT**, press **Return**.
- In the editor, review the file and, if necessary, add any required product specific metadata. Then quit the editor.

3 At a UNIX prompt on the AIT Sun, type **cleartool setview *ViewName***, press **Return**.
- The ***ViewName*** is the name of the view that allows access to the ESDT under CM. For example, type **cleartool setview esdtvob**, press **Return**.

4 At a UNIX prompt on the AIT Sun, type **cd *ESDTpathname***, press **Return**.
- The ***ESDTpathname*** is the full path name of the subdirectory in the ClearCase configured VOB containing the ESDTs for the relevant instrument.
- For example, type **cd /DAAC/ESDT/CER**, press **Return**.

5 At a UNIX prompt on the AIT Sun, type **cp *pathname/DescFilename* .**, press **Return** (note the space and then "dot" at the end of the command).
- The ***pathname*** is the full path name to the directory containing the new ESDT descriptor file.
- The ***DescFilename*** is the file name of the new ESDT descriptor file created in step 1. The file name will be the ESDT's ShortName.
- For example, type **cp /SSIT/CERES/ESDT/CER01aT .**, press **Return** (note the space and then "dot" at the end of the command).

6 At a UNIX prompt on the AIT Sun, type **cleartool checkout -nc .**, press **Return** (note the space and then "dot" at the end of the command).
- This command checks out the current directory (represented by the "dot") from ClearCase. Adding a new file (or element) to a directory represents a modification of the directory. Hence, the directory must be checked out before a file can be checked in.
- The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

7 At a UNIX prompt on the AIT Sun, type **cleartool mkelem -nc *DescFilename***, press **Return.**
- The ***DescFilename*** is the name of the new ESDT descriptor file that was copied over in step 5 and is the file that will be checked into ClearCase. This command creates a ClearCase element from the file in preparation for checking it in. The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the make element step.
- For example, type **cleartool mkelem -nc CER01aT**, press **Return**.

8 At the UNIX prompt on the AIT Sun, type **cleartool checkin -c "comment" *DescFilename***, press **Return**.
- The ***DescFilename*** is the name of the file to be checked into ClearCase. This command performs the check in of the file.
- The **-c *"comment"*** indicates a comment to be associated with the check in. The comment must be enclosed within double quotes and immediately follow the **-c** flag.

- For example, type **cleartool checkin -c "New ESDT generated for the Pre-Release B Testbed" CER01aT**, press **Return**.

9 At the UNIX prompt on the AIT Sun, type **cleartool checkin -nc .**, press **Return** (note the space and then "dot" at the end of the command).

- This command checks in the current directory (represented by the "dot") into ClearCase. The adding of an element (here, a descriptor file) represents a modification to the directory and hence, the new version of the directory must be checked back in. The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the check in step.

Table 5.2.2-2 Generating a New ESDT Descriptor File - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	DescGen *textfile*	press Return
2	vi *DescFilename*	review and then quit editor
3	cleartool setview *ViewName*	press Return
4	cd *ESDTpathname*	press Return
5	cp *pathname*/*DescFilename* .	press Return
6	cleartool checkout -nc .	press Return
7	cleartool mkelem -nc *DescFilename*	press Return
8	cleartool checkin -c "comment" *DescFilename*	press Return
9	cleartool checkin -nc .	press Return

5.3 Registering an ESDT

The tool test_esdt provides the capability to register a new ESDT in the IMF Data Server of the Pre-Release B Testbed. The tool is run from the command line by authorized ECS SSI&T or Systems Administrator personnel. The program creates a new directory for the ESDT in the ECS PDPS configured area. If the ESDT collection is generated for a dynamic science data granule, the PDPS will put into this directory the ESDT descriptor file (with the ESDT's ShortName used as the file name), the science data granule files, and the Target MCFs (*ShortName*.met files) which are used for Inserting the granules to the IMF Data Server. If the ESDT collection is generated for static or permanent data files, the PDPS will put into this directory the ESDT descriptor file (again, with the ESDT's ShortName used as the file name) and the static files. By convention, a container or bucket ESDT is generally created for each PGE to hold all static or permanent files needed by the PGE.

The following list contains the information required for input to the test_esdt tool.

- The archive name which will be supplied by ECS. The archive name, in the IMF, refers to the UNIX path name of the archive.
- The ShortName for the ESDT.

- The ESDT descriptor file name. In the Pre-Release B Testbed IMF, the descriptor file name must be the same as the ShortName.

The Activity Checklist table that follows provides an overview of the process for registering an ESDT. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 5.3-1 Registering an ESDT - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Run the test_esdt tool.	(I) 5.3	
2	SSI&T	Enter the archive name.	(I) 5.3	
3	SSI&T	Enter the ESDT ShortName.	(I) 5.3	
4	SSI&T	Enter the ESDT descriptor file name.	(I) 5.3	
5	SSI&T	Exit the program.	(I) 5.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. A valid descriptor file exists for the ESDT to be registered.
2. The path name to the IMF Data Server archive is known.

To register an ESDT, execute the procedure steps that follow:

1 At the UNIX prompt on an AIT Sun, type **test_esdt**, press **Return**.

2 At the program prompt **Enter archive:**, type ***pathname***, press **Return**.
- The ***pathname*** is the full path name to the IMF archive in which to register the ESDT.
- For example, type **/ECS/Rel_A/PDPS/archive**, press **Return**.
- The environment variable DataServer should contain the full path name to the IMF archive. Type **echo $DataServer**, press **Return** to verify.

3 At the program prompt **Enter ShortName:**, type ***ShortName***, press **Return**.
- The ***ShortName*** is the ShortName of the ESDT (8 characters or less).
- For example, type **CER01aT**, press **Return**.

4 At the program prompt **Enter descriptor:**, type ***DescFilename***, press **Return**.

- The ***DescFilename*** is the file name of the ESDT descriptor file. Include the path name if the descriptor file is not in the current directory.
- For example, type **CER01aT**, press **Return**.
- If the registration is successful, the message “success: ESDT (*ShortName*) added” will be displayed, where *ShortName* will be replaced by the ESDT ShortName just registered.

5 Exit the program.

- Continue on to Section 5.4 to validate successful ESDT registration.

Table 5.3-2 Registering an ESDT - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	test_esdt	press Return
2	*pathname*	press Return
3	*ShortName*	press Return
4	*DescFilename*	press Return
5	exit program	press Return

5.4 Validating Successful ESDT Registration

After registering a new ESDT (see Section 5.3), a subdirectory should be created in the IMF Data Server archive which is located in an ECS PDPS configured area. The new ESDT subdirectory will be named with the ESDT’s ShortName. The ESDT registration tool, test_esdt, will copy the corresponding ESDT descriptor file from the ESC configuration ClearCase VOB to the IMF Data Server archive in the corresponding ESDT subdirectory. When the registration process has completed, the SSI&T staff should validate its success. In the IMF Data Server, the criteria of success will be verifying that the new ESDT subdirectory has been created in the correct directory, that the new ESDT descriptor file has been copied into this subdirectory, and that the contents of the ESDT descriptor file are identical to the ESDT descriptor file in the ClearCase configured VOB.

The Activity Checklist table that follows provides an overview of the process for validating successful registration of the ESDT. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 5.4-1 Validating Successful ESDT Registration - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the IMF Data Server archive.	(I) 5.4	
2	SSI&T	List out the ESDT subdirectories in the archive.	(I) 5.4	
3	SSI&T	Change directories to the subdirectory for the new ESDT.	(I) 5.4	
4	SSI&T	List the contents of the new ESDT subdirectory.	(I) 5.4	
5	SSI&T	Set the ClearCase view to the configured VOB.	(I) 5.4	
6	SSI&T	Compare the contents of the copied ESDT descriptor in the archive to the original in ClearCase.	(I) 5.4	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The path name to the IMF Data Server archive is known.
2. A ClearCase configured VOB for ESDT descriptor files has been properly created and populated.
3. The VA has granted necessary ClearCase privileges to the user.
4. The ESDT descriptor file exists under configuration management in a ClearCase VOB.

To validate successful ESDT registration, execute the procedure steps that follow:

1 At the program prompt on an AIT Sun, type **cd *ArchivePathname***, press **Return**.
- The ***ArchivePathname*** is the full path name of the IMF Data Server archive.
- For example, type **cd /ECS/Rel_A/PDPS/archive**, press **Return**.
- The environment variable DataServer should contain the full path name to the IMF archive. Type **echo $DataServer**, press **Return** to verify.

2 At the program prompt on the AIT Sun, type **ls -al**, press **Return**.
- A list of the current contents will be displayed. These will mostly be directories, one for each ESDT that currently exists. The directories will be named with the ESDT ShortName.

3 At the program prompt on the AIT Sun, type **cd *ShortName***, press **Return**.
- The ***ShortName*** is the name of the subdirectory containing the new ESDT descriptor file.
- For example, type **cd CER01aT**, press **Return**.

4 At the program prompt on the AIT Sun, type **ls -al**, press **Return**.

- A list of the current contents of the ***ShortName*** directory will be displayed.
- For an ESDT newly registered (Section 5.3), the only file listed should be named **.descriptor**. This is the descriptor file for this ESDT.

5 At a UNIX prompt on the AIT Sun, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of the view that allows access to the ClearCase configured VOB. For example, type **cleartool setview esdtvob**, press **Return**.

6 At a UNIX prompt on the AIT Sun, type **diff .descriptor *pathname/ShortName***, press **Return**.

- The **.descriptor** is the file name of the descriptor file for this ESDT. Enter **.descriptor** literally.
- The ***ShortName*** is the file name of the ESDT descriptor file that exists in both the IMF Data Server archive and in the ClearCase configured VOB.
- The ***pathname*** is the full path name to the location in the ClearCase configured VOB where the ESDT descriptor file resides under configuration management.
- For example, type **diff .descriptor /DAAC/ESDT/CER/CER01aT**, press **Return**.
- If no diagnostic messages are displayed, it means that the ESDT registration process was successful. If diagnostic messages are displayed, this indicates some problem with the ESDT registration that should be investigated.

Table 5.4-2 Validating Successful ESDT Registration - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *ArchivePathname*	press Return
2	ls -al	press Return
3	cd *ShortName*	press Return
4	ls -al	press Return
5	cleartool setview *ViewName*	press Return
6	diff .descriptor *pathname/ShortName*	press Return

This page intentionally left blank.

6. Configuration Management

This section describes procedures for handling the configuration management of science software delivered to the DAACs for SSI&T. The COTS tool used for this purpose is ClearCase® by Atria Software, Inc. ClearCase can be run from the command line and via a graphical user interface (GUI).

All data managed under ClearCase are stored in VOBs (Versioned Object Bases), which are the "public" storage areas and views, which are the "private" storage areas. VOBs are data structures that can only be created by a VOB administrator using the mkvob ("make vob") command. They are mounted as a file system and when viewed through a view, VOBs appear as standard UNIX directory tree structures. This file system, accessed through its mount point, has a version-control dimension which contains file elements and versions of file elements. ClearCase maintains multiple versions of a file organized into a hierarchical version tree which may include many branches. For the purposes of SSIT, it is suggested that branches not be created in managing the files of the science software. All versions of the software files should remain on their main branch of the tree and therefore easily accessed by the default configuration of any view that may be set by the user.

Data that are under configuration management in ClearCase are said to be *checked in*. In order to alter a checked in data element (*e.g.* a file) to make a newer version of it, the data element must first be *checked out*. Once the change has been made to the checked out version, it is checked in again. The VOB will then contain both versions of the data element (in this case, a file) and either can be retrieved at a later time.

In general, executable binary files, object files, and data files should not be checked into ClearCase. Binary and object files are not stored efficiently in ClearCase; data files for science software may be extremely large (multiple gigabytes) and a VOB is typically not sized for this. Files that should be checked into ClearCase include source code, scripts, makefiles, assorted build and run scripts, documentation, and other ASCII files.

The administrator in charge of the VOB is referred to as the VOB administrator (VA).

All ClearCase procedures assume that the user's umask is set to 002.

6.1 Creating a View in ClearCase

In order to make files and directories that are in a ClearCase VOB (Versioned Object Base) visible and accessible, a ClearCase view must be set. This procedure describes how to create a ClearCase view to be used later (in Section 6.2).

A ClearCase view need only be created once. Once created, the view can be set at the beginning of each user session. Multiple views for a single user may be created.

In order for the SSI&T tools under the SSIT Manager to have access to the ClearCase VOB, the ClearCase view must be set *before* the SSIT Manager is run.

The Activity Checklist table that follows provides an overview of the process for creating a ClearCase view. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 6.1-1 Creating a View in ClearCase - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Create the ClearCase view.	(I) 6.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ClearCase VOB has been properly created and populated.
2. The VA has granted ClearCase privileges for the user.

To create a ClearCase view, execute the procedure steps that follow:

1 At a UNIX prompt, type **cleartool mkview -tag *ViewName ViewPath*/*ViewName*.vws**, press **Return**.

- The ***ViewPath*** is the full path name to the directory where ClearCase views are stored. This information should be supplied by the SA. A typical example is /net/mssg1sungsfc/viewstore/.
- The ***ViewName*** is the name of the view being created (typically set to the user id, for example, jdoe). View names should have .vws as the file name extension.
- For example, to create a view for user jdoe, type **cleartool mkview -tag jdoe /net/mssg1sungsfc/data/viewstore/jdoe.vws**, press **Return**.
- Multiple views can be created in the same manner for the same user. Each view, however, must have a unique name. For example, jdoe_test and jdoe_devel.

Table 6.1-2 Creating a View in ClearCase - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool mkview -tag *ViewName ViewPath*/*ViewName*.vws	press Return

6.2 Setting a View in ClearCase

In order to make files and directories that are in a ClearCase VOB (Versioned Object Base) visible and accessible, a ClearCase view must be set. Only one view can be set (active) at a time.

Section 6.1 describes how to create a ClearCase view.

The Activity Checklist table that follows provides an overview of the process for setting a ClearCase view. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 6.2-1 Setting a View in ClearCase - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view.	(I) 6.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ClearCase VOB has been properly created and populated.
2. A ClearCase view has been created for the user (see Section 6.1).

To set a ClearCase view, execute the procedure steps that follow:

1 At a UNIX prompt, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of the view to be set. For example, type **cleartool setview jdoe**, press **Return**.
- The ***ViewName*** may be a view that another user has created and owns. The restriction, in this case, is that files cannot be checked in.

Table 6.2-2 Setting a View in ClearCase - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool setview *ViewName*	press Return

6.3 Importing Multiple Files into ClearCase from a Directory Structure

This procedure explains how to place the entire contents of a UNIX directory structure under ClearCase. A UNIX directory structure refers to all the files and subdirectories under some top-level directory.

This procedure is geared toward science software deliveries. In such cases, science software is delivered is in the form of a UNIX *tar* files. A *tar* file has been unpacked (un*tar*-red) and the contents are to be placed under ClearCase configuration management.

The Activity Checklist table that follows provides an overview of the process for importing multiple files into ClearCase from a directory structure. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 6.3-1 Importing Multiple Files into ClearCase from a Directory Structure - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the parent directory of the UNIX directory structure	(I) 6.3	
2	SSI&T	Create a conversion script.	(I) 6.3	
3	VA	Run script to place all elements under ClearCase	(I) ???	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ClearCase VOB has been properly created.
2. A ClearCase view is **not** required to perform this procedure.

To import multiple files and/or directories into ClearCase from a UNIX directory structure, execute the procedure steps that follow:

1 At a UNIX prompt, type **cd *ParentPathname***, press **Return**

- The ***ParentPathname*** is the path name of the directory that *contains* the directory structure to be brought into ClearCase. This is *not* the VOB. For example, to bring the pge2 directory which sits under /MOPITT (i.e. its path name is /MOPITT/pge2) into ClearCase along with all files and subdirectories below it, **cd /MOPITT**, press **Return**.

2 At the UNIX prompt, type **clearcvt_unix -r *DirName***, press **Return**.

- The ***DirName*** is the name of the directory in which it and everything below it is to be brought into ClearCase. For example, based on the example in step 1, **clearcvt_unix -r pge2**, press **Return**. A conversion script will be then be created. The **-r** causes all subdirectories to be recursively included in the script created.

3 Contact the VA and request that the utility script **cvt_script** be run on the script created in step 2.

- The VOB administrator (VA) is the only one who can run the **cvt_script** because it modifies the VOB.

Table 6.3-2 Importing Multiple Files into ClearCase from a Directory Structure - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *ParentPathname*	press Return
2	clearcvt_unix -r *DirName*	press Return
3	(No entry)	request VA run cvt_script

6.4 Entering a Single File into ClearCase

This procedure explains how to put a single file under configuration management using ClearCase.

The Activity Checklist table that follows provides an overview of the process for entering a single file into ClearCase. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 6.4-1 Entering a Single File into ClearCase - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view.	(I) 6.4	
2	SSI&T	Go to the VOB directory in which the file should be checked in.	(I) 6.4	
3	SSI&T	Check out the directory.	(I) 6.4	
4	SSI&T	Create a new ClearCase element.	(I) 6.4	

Table 6.4-1 Entering a Single File into ClearCase - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
5	SSI&T	Check in the new file.	(I) 6.4	
6	SSI&T	Check back in the directory.	(I) 6.4	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ClearCase VOB has been properly created.
2. A ClearCase view exists for the user through which the desired VOB directory can be seen.

To enter a single file into ClearCase, execute the procedure steps that follow:

1 At a UNIX prompt, type **cleartool setview *ViewName***, press **Return**
- The ***ViewName*** is the name of the ClearCase view (see Section 6.1 to create a view). For example, **cleartool setview jdoe,** press **Return**.

2 At the UNIX prompt, type **cd *pathname***, press **Return**.
- The ***pathname*** is the full path name of the subdirectory in the VOB into which the file is to be checked in. For example, to check a file into the VOB directory /VOB1/pge2/scripts/, type **cd /VOB1/pge2/scripts/**, press **Return**. If the desired directory cannot be seen, it could mean that the view has not been set or the properties of the view do not allow the directory to be seen; check with the VA.

3 At a UNIX prompt, type **cp *pathname/filename* .**, press **Return** (note the space and then "dot" at the end of the command)**.**
- The ***pathname*** is the full path name to the directory where the file to be checked in exists and ***filename*** is the file name of the file to be checked in. This command copies a file over into the VOB area in preparation for checking it in. For example, to copy over a file named MISR_calib.c in directory /pge/pge34/ to be checked in, type **cp pge/pge34/MISR_calib.c .**, press **Return** (again, note the space and then "dot" at the end of the command).

4 At the UNIX prompt, type **cleartool checkout -nc .**, press **Return** (note the space and then "dot" at the end of the command).
- This command checks out the current directory (represented by the "dot") from ClearCase. Adding a new file (or element) to a directory represents a modification of the directory. Hence, the directory must be checked out before a file can be checked in.

5 At a UNIX prompt, type **cleartool mkelem -nc *filename***, press **Return.**

- The ***filename*** is the name of the file that was copied over in step 4 and is the file that will be checked into ClearCase. This command creates a ClearCase element from the file in preparation for checking it in. The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the make element step.

6 At the UNIX prompt, type **cleartool checkin -nc *filename***, press **Return**.
- The ***filename*** is the name of the file to be checked into ClearCase. This command performs the check in of the file. The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the check in step.

7 At the UNIX prompt, type **cleartool checkin -nc .**, press **Return** (note the space and then "dot" at the end of the command).
- This command checks in the current directory (represented by the "dot") into ClearCase. The adding of an element (here, a file) represents a modification to the directory and hence, the new version of the directory must be checked back in. The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the check in step.

Table 6.4-2 Entering a Single File into ClearCase - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool setview *ViewName*	press Return
2	cd *pathname*	press Return
3	cp *pathname/filename* .	press Return
4	cleartool checkout -nc .	press Return
5	cleartool mkelem -nc *filename*	press Return
6	cleartool checkin -nc *filename*	press Return
7	cleartool checkin -nc .	press Return

6.5 Entering a New Directory into ClearCase

This procedure explains how to put a new directory (empty) into ClearCase.

The assumptions are that a VOB exits and is mounted at a known UNIX directory. A ClearCase view exists for the SSI&T operator.

The Activity Checklist table that follows provides an overview of the process for entering a new directory into ClearCase. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 6.5-1 Entering a New Directory into ClearCase - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view.	(I) 6.2	
2	SSI&T	Go to the VOB directory in which the new directory is to be added.	(I) 6.5	
3	SSI&T	Check out the parent directory.	(I) 6.5	
4	SSI&T	Create the new directory.	(I) 6.5	
5	SSI&T	Check in the new directory.	(I) 6.5	
6	SSI&T	Check back in the parent directory.	(I) 6.5	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ClearCase VOB has been properly created.
2. A ClearCase view exists for the user through which the desired VOB parent directory can be seen.

To enter a single file into ClearCase, execute the procedure steps that follow:

1 At a UNIX prompt, type **cleartool setview *ViewName***, press **Return**
- The ***ViewName*** is the name of the ClearCase view (see Section 6.1 to create a view). For example, **cleartool setview jdoe,** press **Return**.

2 At the UNIX prompt, type **cd *pathname***, press **Return**.
- The ***pathname*** is the full path name of the parent directory in the VOB in which the new directory is to be added. For example, if a new directory is to be added under /VOB1/pge4, type **cd /VOB1/pge4**, press **Return**.

3 At the UNIX prompt, type **cleartool checkout -nc .**, press **Return** (note the space and then “dot” at the end of the command).
- This command checks out the current directory (represented by the “dot”) from ClearCase. This directory will be the parent directory of the new directory. Adding a new directory (or element) to another directory represents a modification of the directory. Hence, the directory must be checked out before a new directory can be added and checked in.

4 At a UNIX prompt, type **cleartool mkdir -nc *DirectoryName***, press **Return.**
- The ***DirectoryName*** is the name of the new directory being created. This command creates the new directory.
- The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the directory creation step.

5 At the UNIX prompt, type **cleartool checkin -nc *DirectoryName***, press **Return**.

- The ***DirectoryName*** is the name of the new directory created in step 4. This command performs the check in of the new directory. The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check in step.

6 At the UNIX prompt, type **cleartool checkin -nc .**, press **Return** (note the space and then “dot” at the end of the command).

- This command checks in the current directory (represented by the “dot”) into ClearCase. The adding of an element (here, a directory) represents a modification to the current directory and hence, the new version of the directory must be checked back in.
- The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check in step.

Table 6.5-2 Entering a New Directory into ClearCase - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool setview *ViewName*	press Return
2	cd *pathname*	press Return
3	cleartool checkout -nc .	press Return
4	cleartool mkdir -nc *DirectoryName*	press Return
5	cleartool checkin -nc *DirectoryName*	press Return
6	cleartool checkin -nc *.*	press Return

6.6 Checking Out an Element From ClearCase

This procedure explains how to check out an element from ClearCase so that it can be modified. An element refers to a directory or file in ClearCase, that is, under configuration management. Modifications made to a file or directory cannot be saved in ClearCase unless the file or directory had been checked out first.

The assumptions are that a VOB exits and is mounted at a known UNIX directory. A ClearCase view exists for the SSI&T operator.

The Activity Checklist table that follows provides an overview of the process for checking out an element from ClearCase. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 6.6-1 Checking Out an Element From ClearCase - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view.	(I) 6.2	
2	SSI&T	Check out the element (file or directory).	(I) 6.6	
3	SSI&T	If desired, cancel the check out.	(I) 6.6	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ClearCase VOB has been properly created.
2. A ClearCase view exists for the user through which the desired VOB element can be seen. See Section 6.7 for how to check in a modified element.

To enter a single file into ClearCase, execute the procedure steps that follow:

1 At a UNIX prompt, type **cleartool setview *ViewName***, press **Return**

- The ***ViewName*** is the name of the ClearCase view (see Section 6.1 to create a view). For example, **cleartool setview jdoe,** press **Return**.

2 At the UNIX prompt, type **cleartool checkout -nc *element***, press **Return**.

- The ***element*** is the name of the file or directory (full path name allowed) that is to be checked out (and later modified). The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

3 This step is optional; it is performed to cancel a check out. At a UNIX prompt, type **cleartool uncheckout -nc *element***, press **Return.**

- The ***element*** is the name of the file or directory (full path name allowed) checked out (as in step 2, above). This command cancels the check out of an element. The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the uncheck out step. Note that ClearCase will not allow an unmodified element to be checked in. Instead, the uncheck out command must be used.

Table 6.6-2 Checking Out an Element From ClearCase - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool setview *ViewName*	press Return
2	cleartool checkout -nc *element*	press Return
3	(Optional) cleartool uncheckout -nc *element*	press Return

6.7 Checking a Modified Element into ClearCase

This procedure explains how to check in a modified element to ClearCase. An element refers to a directory or file in ClearCase, that is, under configuration management. Modifications made to a file or directory cannot be saved in ClearCase unless the file or directory had been checked out first. See Section 6.6 on how to check out an element.

The assumptions are that a VOB exits and is mounted at a known UNIX directory. A ClearCase view exists for the SSI&T operator.

The Activity Checklist table that follows provides an overview of the process for checking a modified element into ClearCase. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 6.7-1 Checking a Modified Element into ClearCase - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view.	(I) 6.2	
2	SSI&T	Check in the modified element (file or directory).	(I) 6.7	
3	SSI&T	If element was not modified, cancel the check out.	(I) 6.7	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ClearCase VOB has been properly created.
2. A ClearCase view exists for the user through which the desired VOB element can be seen. See Section 6.6 for how to check out an element before modifying.

To enter a single file into ClearCase, execute the procedure steps that follow:

1 At a UNIX prompt, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of the ClearCase view (see Section 6.1 to create a view). For example, **cleartool setview jdoe**, press Return.

2 At the UNIX prompt, type **cleartool checkin -nc *element***, press **Return**.

- The ***element*** is the name of the file or directory (full path name allowed) that is to be checked out (and later modified). The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

3 This step is optional; it is performed when ClearCase does not accept a check in because the element was not modified. In this case, the check out must be canceled. At a UNIX prompt, type **cleartool uncheckout -nc *element***, press **Return.**

- The ***element*** is the name of the file or directory (full path name allowed) checked out. This command cancels the check out of an element. The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the uncheck out step.

Table 6.7-2 Checking a Modified Element into ClearCase - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool setview *ViewName*	press Return
2	cleartool checkin -nc *element*	press Return
3	(Optional) cleartool uncheckout -nc *element*	press Return

7. The SSIT Manager

The principal tool used during SSI&T is the SSIT Manager. The SSIT Manager is the top-level graphical user interface (GUI) environment presented to SSI&T personnel. Its purpose is to bring together the tools needed for SSI&T into a single, graphical environment.

7.1 General Set Up of the SSIT Manager

The SSIT Manager requires a configured environment within which to run; it runs only on the Release A AIT Suns. The set up steps described in this section need only be done the first time a SSI&T operator uses the SSIT Manager.

Note that a prior condition for using the SSIT Manager and all SSIT tools is that the install script DpAtINSTALL.sh has been run at system level. This is normally done each time the ECS PDPS/SSIT software is redelivered (if any). A simple check that this operation has been performed is to check the date on file /usr/ecs/Rel_A/CUSTOM/bin/DPS/DpAtEnv.csh . The date should be the consistent with the latest delivery date of the ECS PDPS/SSIT software. If it is not then the system administrator must be informed and the install script run before SSIT tools may be used.

The Activity Checklist table that follows provides an overview of the process for the general set up of the SSIT Manager. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 7.1-1 General Set Up of the SSIT Manager - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Copy the default Process Control File to the home directory.	(I) 7.1	
2	SSI&T	Bring up the .cshrc file in a text editor.	(I) 7.1	
3	SSI&T	Add line setting the DPATMGR_HOME environment variable.	(I) 7.1	
4	SSI&T	Add line setting the PGS_PC_INFO_FILE environment variable.	(I) 7.1	
5	SSI&T	Add line setting the PGSHOME environment variable.	(I) 7.1	

Table 7.1-1 General Set Up of the SSIT Manager - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
6	SSI&T	Add line setting the PGSMSG environment variable.	(I) 7.1	
7	SSI&T	Add line sourcing the DpAtEnv.csh file.	(I) 7.1	
8	SSI&T	Optionally, set the DISPLAY environment variable.	(I) 7.1	
9	SSI&T	Save the file and exit the editor.	(I) 7.1	
10	SSI&T	Reinitialize the shell environment.	(I) 7.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative like T shell) is the current command shell.
2. A file named .cshrc exists in the operator's home directory.
3. The operator is knowledgeable in the use of text editors (such as *vi* or *emacs*).

To set up the environment for the SSIT Manager, execute the procedure steps that follow. Note that in the following, EcsCustomSw is nominally in the directory /usr/ecs/Rel_A/CUSTOM on the AIT Sun:

1 At a UNIX prompt on an AIT Sun, type **cp *SSITpcfPathname*/*filename* $HOME/*mySSITpcf***, press **Return**.
- The ***SSITpcfPathname*** is the full path name of the location of the SSIT Manager's default internal Process Control File (PCF).
- The ***filename*** is the file name of the PCF.
- The complete path and file name is usually EcsCustomSw/DpAtMgrInternal.pcf*machine_name*, where *machine_name* is the name of the machine on which the SSIT software was installed
- The ***mySSITpcf*** is the file name to the private copy of the PCF that the SSI&T operator will use when running the SSIT Manager. The **$HOME** is the environment variable for the user's home directory. For example, **cp /RelA/data/PCF.v5.ssit.dprs1gsfc $HOME/myPCF**, press **Return**.

2 At a UNIX prompt on the AIT Sun, type **vi $HOME/.cshrc**, press **Return.**
- This command invokes the *vi* editor and reads in the .cshrc file from the user's home directory.
- Any text editor may be used such as *emacs*. For example, **emacs $HOME/.cshrc**, press **Return**.

3 In the file, add the following line if not already there: **setenv DPATMGR_HOME *SSITmanagerPathname***.

- The ***SSITmanagerPathname*** is the full path name to the home directory of the SSIT Manager on the AIT Sun. This information should be provided by the SA.

4 In the file, add the following line if not already there: **setenv PGS_PC_INFO_FILE $HOME/*mySSITpcf***
- The ***mySSITpcf*** is the full path name to the private copy of the PCF to be used with the SSIT Manager when you run it (from step 1).

5 In the file, add the following line if not already there: **setenv PGSHOME *ToolkitPathname***
- The ***ToolkitPathname*** is the full path name to location of the SDP Toolkit home directory on the AIT Sun. This information should be provided by the SA.

6 In the file, add the following line if not already there: **setenv PGSMSG $PGSHOME/message**
- The line must be after the line entered in step 5.

7 In the file, add the following line if not already there: **source $DPATMGR_HOME/bin/sun5/DpAtEnv.csh**
- This command causes the file DpAtEnv.csh to be run (sourced) and other environment variables to be set.

8 This step is optional. In the file, add the following line if not already there: **setenv DISPLAY *machinename*:0.0**
- The ***machinename*** is the name of the machine on which the SSIT Manager is to be displayed and operated. For example, if the machine name is spr1sgigsfc, then enter **setenv DISPLAY spr1sgigsfc**.
- If access to the SSIT Manager will not be from the same machine (all or most of the time), this command should not be added to the .cshrc. Instead, the operator will have to run the command from the command line at the UNIX prompt each time *before* the SSIT Manager is started.

9 Save the changes made to the .cshrc file and exit the editor.
- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor’s documentation.

10 At the UNIX prompt on the AIT Sun, type **source $HOME/.cshrc**, press **Return**.
- This command causes the .cshrc file to get run and the new commands in it to be executed.
- Optionally, the user may logout and then log back in. The result will be the same as above.

Table 7.1-2 General Set Up of the SSIT Manager - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cp *SSITpcfPathname*/*filename* $HOME/*mySSITpcf*	press Return
2	vi $HOME/.cshrc	press Return
3	setenv DPATMGR_HOME *SSITmanagerPathname*	add line to .cshrc file
4	setenv PGS_PC_INFO_FILE $HOME/*mySSITpcf*	add line to .cshrc file
5	setenv PGSHOME *ToolkitPathname*	add line to .cshrc file
6	setenv PGSMSG $PGSHOME/message	add line to .cshrc file
7	source $DPATMGR_HOME/bin/sun5/DpAtEnv.csh	add line to .cshrc file
8	(Optional) setenv DISPLAY *machinename*:0.0	add line to .cshrc file
9	Save and quit the editor	invoke editor command
10	source $HOME/.cshrc	press Return

7.2 Set Up of a Checklist for the SSIT Manager

The SSIT Manager offers the capability of maintaining user-defined checklist of SSI&T activities. The checklist is presented in the main window of the SSIT Manager. This procedure explains how to set up this checklist.

The Activity Checklist table that follows provides an overview of the process for setting up a checklist for the SSIT Manager. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 7.2-1 Set Up of a Checklist for the SSIT Manager - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Start an xterm session on an AIT Sun.	(I) 7.2	
2	SSI&T	Make a private copy of the sample checklist file.	(I) 7.2	
3	SSI&T	Invoke editor with private checklist file.	(I) 7.2	
4	SSI&T	Set path name to SSIT Manager’s database.	(I) 7.2	
5	SSI&T	Add checklist items to file.	(I) 7.2	
6	SSI&T	Save the file and exit the editor.	(I) 7.2	
7	SSI&T	If necessary, remove old database files.	(I) 7.2	
8	SSI&T	Bring up the private copy of the SSIT Manager’s PCF in a text editor.	(I) 7.2	

Table 7.2-1 Set Up of a Checklist for the SSIT Manager - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
9	SSI&T	Edit line pointing to checklist file.	(I) 7.2	
10	SSI&T	Save the file and exit the editor.	(I) 7.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager has been properly installed. This can be verified by running the SSIT Manager; the sample (default) checklist should be displayed in the main window of the GUI.
2. A private copy of the Process Control File (PCF) used by the SSIT Manager has been created in the user's home directory (see Section 7.1).

See Section 7.1 for general set up of the SSIT Manager; see Section 7.3 for routine running of the SSIT Manager. See Appendix B for an example of a complete SSI&T checklist.

To create a user-defined checklist for the SSIT Manager, execute the procedure steps that follow:

1 At a UNIX prompt on an AIT Sun, type **xterm &**, press **Return**
- This command starts a new xterm session on the AIT Sun.

2 At the UNIX prompt on the AIT Sun, type **cp $DPATMGR_HOME/data/checklist.sample $HOME/*mychecklist***, press **Return**.
- The ***mychecklist*** is the file name for a private copy of the checklist file. It will be this copy that will be edited in the steps which follow.
- **$DPATMGR_HOME** is an environment variable pointing to the home directory of the SSIT Manager.
- **$HOME** is the user's home directory. Although the checklist file does not have to be in the user's home directory, the following steps assume that it is.

3 At a UNIX prompt on the AIT Sun, type **vi $HOME/*mychecklist***, press **Return.**
- The ***mychecklist*** is the file name of the checklist created in step 1. This command invokes the *vi* editor and reads in the checklist file from the user's home directory. Alternatively, any text editor may be used such as *emacs*. For example, **emacs $HOME/*mychecklist***, press **Return**.

4 In the file, search for the lines of the form **DATABASE=*ssitUserPathname*/*filename*** and **CHECKLIST=*title***. Edit these lines.
- The ***ssitUserPathname*** is the full path name to where the database files (used in conjunction with the checklist) will be placed. Typically, this is the user's home directory.

- The ***filename*** is the base name for the database files. Two database files will be created in the directory given by ***ssitUserPathname*** using this base name. They will be ***filename***.dir and ***filename***.pag. For example, using **DATABASE=/home/jdoe/CEREScheckIist** will result in two database files, **CEREScheckIist.dir** and **CEREScheckIist.pag**, being created in **/home/jdoe/**.
- The ***title*** is a user-selected name that will appear in the SSIT Manager GUI above the checklist items in the main window.

5 In the file, add the items that will appear in the checklist. Search for lines in the file of the form **ITEM=*ChecklistStep***. Each of these lines specifies a procedure or step taken during SSI&T. Edit these lines and/or add lines.

- The ***ChecklistStep*** is any text string (without quotes). Typically, each ***ChecklistStep*** is a SSI&T operational procedure or task. The source of these steps may be existing documentation, for example, the SSI&T Agreement between the Instrument Team and the DAAC.
- Add **ITEM** lines as necessary. Checklist items will appear in the SSIT Manager window in the order entered in this file.
- There is no limitation to the number of **ITEM** lines that can be entered. However, a large checklist will result in the SSIT Manager taking a significantly longer time to start up.

6 Save the changes made to the checklist file and exit the editor.

- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor’s documentation.

7 If database files already exist from a previous checklist set up, at the UNIX prompt on the AIT Sun, type **rm *ssitUserPathname/filename*.dir** press **Return**. Type **rm s*sitUserPathname/filename*.pag**, press **Return**.

- The ***ssitUserPathname*** and ***filename*** are the full path name and file name set in the checklist file in step 4.
- If these database files already exist (from a previous checklist set up), an error would result when the SSIT Manager was started.

8 At a UNIX prompt on the AIT Sun, type **vi $HOME/*mySSITpcf***, press **Return**.

- The ***mySSITpcf*** is the file name of the private copy of the PCF used by the SSIT Manager (refer to Section 7.1, step 1).

9 In the file, search for identifier 603 beginning in the first column. The line should be of the form: **603|DpAtMgrLogDatabaseInit|*ssitUserPathname*/*filename***. Edit this line.

- The ***ssitUserPathname*** and ***filename*** are the full path name and file name set in the checklist file in step 4.

10 Save the changes made to the Process Control File and exit the editor.

- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor’s documentation.
- The set up of the SSIT Manager’s checklist is now complete.

Table 7.2-2 Set Up of a Checklist for the SSIT Manager - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	xterm &	press Return
2	cp $DPATMGR_HOME/data/checklist.sample $HOME/*mychecklist*	press Return
3	vi $HOME/*mychecklist*	press Return

4	DATABASE=*ssitUserPathname*/ *filename*
	CHECKLIST=*title*
		edit in checklist file

5	ITEM= *ChecklistStep*	add/edit in checklist file
6	Save and quit the editor	invoke editor command
7	(If necessary) rm *ssitUserPathname/filename*.dir rm *ssitUserPathname/filename*.pag	 press Return press Return
8	vi $HOME/*mySSITpcf*	press Return
9	603|DpAtMgrLogDatabaseInit|*ssitUserPathname/ filename*	edit in PCF
10	:wq	press Return

7.3 Running the SSIT Manager

This procedure describes the routine running of the SSIT Manager.

The Activity Checklist table that follows provides an overview of the process for running the SSIT Manager. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 7.3-1 Running the SSIT Manager - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into an AIT Sun.	(I) 7.3	
2	SSI&T	Set the DISPLAY environment variable.	(I) 7.3	
3	SSI&T	If necessary, set a ClearCase view.	(I) 6.2	
4	SSI&T	Start the SSIT Manager	(I) 7.3	
5	SSI&T	Optionally, dump the checklist database log.	(I) 7.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager has been properly installed.
2. Although not required, it is assumed that a customized checklist is used by the SSIT Manager and that a private copy of the PCF has an entry that points to it.
3. The user's shell is C shell or a derivative of C shell (*e.g.* T shell).

See Section 7.1 for general set up of the SSIT Manager; see Section 7.2 for setting up the SSIT Manager's checklist file. See Section 6.1 for creating a ClearCase view and Section 6.2 for using a ClearCase view.

To routinely run the SSIT Manager, execute the procedure steps that follow:

1 If not already on an AIT Sun, log into one from your machine.

2 **A**t the UNIX prompt on an AIT Sun from which the SSIT Manager is to be run, type **setenv DISPLAY *hostname*:0.0**, press **Return**.

- The ***hostname*** is the name of the machine on which the SSIT Manager is to be displayed, *i.e.* the machine that your are using.
- If the machine your are on is an AIT Sun, this step should not be necessary; the variable should be set by default to **:0.0**, meaning the machine that you are on.
- As explained in Section 7.1, step 8, this command could be placed in the .cshrc file so that it is automatically set at login.
- To verify a setting, type **echo $DISPLAY**, press **Return**.

3 If necessary, at the UNIX prompt on an AIT Sun from which the SSIT Manager is to be run, type **cleartool setview *ViewName***, press **Return.**

- The ***ViewName*** is the ClearCase view to be used while the SSIT Manager is running in this session. For example, type **cleartool jdoe**, press **Return**.
- A ClearCase view is required only if the SSIT Manager needs to be able to "see" into a ClearCase VOB; a view is not necessary otherwise.

4 At the UNIX prompt on the AIT Sun, type **DpAtMgr ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/DpAtMG_*daac*.CFG &**, press **Return**.

- The ***daac*** is one of {GSFC, EDC, LARC, NSIDC} .
- The **&** (ampersand) causes the SSIT Manager to be run as a background process, freeing up the UNIX prompt.
- For example, type **DpAtMgr ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/DpAtMG_GSFC.CFG**, press **Return**.
- Various messages from the SSIT Manager will appear in this window as it is running. For this reason, avoid using this window for other tasks until the SSIT Manager has terminated.

5 Optionally, at the UNIX prompt on the AIT Sun, type **DpAtMgrLogDump *ssitUserPathname/filename***, press **Return**.

- The ***ssitUserPathname*** and ***filename*** is the full path name and filename used in the **DATABASE=** line of the checklist file being used. See Section 7.2, step 4. For example, if the **DATABASE=** line in the checklist file was **DATABASE=/home/jdoe/CERESchecklist**, then type **DpAtMgrLogDump /home/jdoe/CERESchecklist**, press **Return**.
- The contents of the database will be displayed on the screen. To redirect the output to a file, type **DpAtMgrLogDump *ssitUserPathname/filename* > *outputfile***, press **Return**. The ***outputfile*** is the file name to be given to the output file.

Table 7.3-2 Running the SSIT Manager - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	(No entry)	log onto AIT Sun
2	setenv DISPLAY *hostname*:0.0	press Return
3	(If necessary) cleartool setview *ViewName*	press Return
4	DpAtMgr ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/DpAtMG_*daac*.CFG &	press Return
5	(Optional) DpAtMgrLogDump *ssitUserPathname/filename*	press Return

This page intentionally left blank.

8. Standards Checking of Science Software

All science software delivered to the DAACs to be run in the ECS must comply with the NASA Earth Science Data and Information Systems (ESDIS) Project standards. The Project standards and guidelines are contained in the document *Data Production Software and Science Computing Facility (SCF) Standards and Guidelines, Revision A, October 1996* (423-16-01).

The currently standards mandate that all science software be written in C, FORTRAN 77, Fortran 90, or Ada. Allowed script languages include C shell, Bourne shell, Korn shell, and Perl. The standards for C, FORTRAN 77, and Fortran 90 are in general ANSI compliance. Certain extension, particularly in FORTRAN 77, are allowed by the guidelines.

In addition, the ESDIS standards and guidelines document mandates that certain functions are prohibited from science software. These are generally functions (or script utilities) that are a problem for the ECS.

8.1 Checking for ESDIS Standards Compliance in FORTRAN 77

This procedure describes how to use the COTS tool FORCHECK to science software written in FORTRAN 77 for ESDIS standards compliance.

FORCHECK can be accessed from the SSIT Manager, however, this procedure assumes that FORCHECK will be run from the UNIX prompt using a script.

The Activity Checklist table that follows provides an overview of the process for checking ESDIS standards compliance in FORTRAN 77. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.1-1 Checking for ESDIS Standards Compliance in FORTRAN 77 - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into an AIT Sun.	(I) 8.1	
2	SSI&T	If necessary, compile SMF files.	(I) 9.3	
3	SSI&T	Create a script that contains commands to run FORCHECK.	(I) 8.1	
4	SSI&T	Give the script execute permissions.	(I) 8.1	

Table 8.1-1 Checking for ESDIS Standards Compliance in FORTRAN 77 - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
5	SSI&T	Set the ClearCase view.	(I) 8.1	
6	SSI&T	Run the FORCHECK script.	(I) 8.1	
7	SSI&T	Examine the FORCHECK output file.	(I) 8.1	
8	SSI&T	Examine the FORCHECK list file.	(I) 8.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The FORTRAN 77 science software source code is available, accessible, and has read permissions for the user.
2. The environment variable FCKCNF has been set to the location of the FORCHECK configuration file.
3. The C shell (or a derivative) is the current command shell.

FORCHECK is available only on the AIT Suns.

To check for ESDIS standards compliance in FORTRAN 77 code, execute the procedure steps that follow:

1 If not already on an AIT Sun, log into one from your machine.

2 If necessary, compile status message facility (SMF) files (see Section 9.3).
- This step is necessary to produce the include file required by the PGE.

3 At the UNIX prompt on the AIT Sun, type **vi *RunFile***, press **Return**.
- The ***RunFile*** is the file name to of the FORCHECK run script to be created. Any text editor may be used for this procedure step. The form of this script is shown in the template below:

```
forchk -l listfile.lst \
-I :incdir1:incdir2:[ … ] \
/pathname/source_code_1.f \
/pathname/source_code_2.f \
         .
         .
         .
/pathname/source_code_n.f
```

- The ***RunFile*** is the file name to of the FORCHECK run script being created.

- The first line of ***RunFile*** should only contain the line: `forchk -l` ***ListFile***`.lst \`, as shown in the template above. The ***ListFile*** is the name to be given to the list output that will be produced. This file contains the source listing with messages produced by FORCHECK. Note that the terminal back slash (\) is required.
- The second line of ***RunFile*** contains the list of include (header file) directories. Each directory should be the full path name and each should be separated by a colon (:) only. A colon by itself should be placed at the beginning of the list (as shown in the template); this represents the current directory.
- Typical include directories to be used on the second line will be the include directories for the SDP Toolkit, HDF, SMF include files, and the science software itself.
- The second line should end in a terminal back slash (\).
- The third and subsequent lines should contain the full path names and file names of each FORTRAN 77 source file to be checked. Place each source file on a separate line. Terminate each line with a back slash (\) **except** the last line.
- Note that *all* lines in the ***RunFile*** end with a back slash (\) **except** the last one.
- Once completed, save the file and exit the editor.
- Do **not** use environment variables in the ***RunFile*** (*e.g.* PGSINC, HDFINC). They will not work.

4 At the UNIX prompt on the AIT Sun, type **chmod +x *RunFile***, press **Return**.
- The ***RunFile*** is the file name of the FORCHECK run script created in step 2.
- This command changes the file's disposition to "execute" allowing it to be run by invoking its name.

5 If required, at the UNIX prompt on the AIT Sun, type **cleartool setview *ViewName***, press **Return**.
- The ***ViewName*** is the name of a view allowing the FORTRAN 77 source files to be accessible.
- This step is only necessary if any of the FORTRAN 77 source files are in ClearCase (in the VOB under configuration management).

6 At the UNIX prompt on the AIT Sun, type ***RunFile* > & *FORCHECKoutput***, press **Return**.
- The ***RunFile*** is the file name of the FORCHECK run script created in step 2.
- The ***FORCHECKoutput*** is the file name for the output file produced. Note that this output file differs from the ***ListFile*** specified in the FORCHECK run script.
- The **>&** is a C shell construct that causes standard error (where the output from FORCHECK normally emerges) to be redirected to a file.

7 At the UNIX prompt on the AIT Sun, type **vi *FORCHECKoutput***, press **Return**.
- The ***FORCHECKoutput*** is the file name for the output file produced in step 5.
- The ***FORCHECKoutput*** file will contain any warnings, errors, and other messages from FORCHECK. A summary will be at the bottom of the file.

- Any text editor may be used for this procedure step.

8 At the UNIX prompt on the AIT Sun, type **vi** ***ListFile***, press **Return**.

- The ***ListFile*** is the file name for the list file specified on line 1 in the FORCHECK run script.
- The ***ListFile*** file will contain FORCHECK messages similar to the ***FORCHECKoutput*** file embedded in the source code listing.
- Any text editor may be used for this procedure step.

Table 8.1-2 Checking for ESDIS Standards Compliance in FORTRAN 77 - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	(No entry)	log onto AIT Sun
2	(If necessary) (No entry)	compile SMF files
3	vi *RunFile*	press Return then use template to create and save file
4	chmod +x *RunFile*	press Return
5	(Optional) cleartool setview *ViewName*	press Return
6	*RunFile* >& *FORCHECKoutput*	press Return
7	vi *FORCHECKoutput*	press Return
8	(Optional) vi *ListFile*	press Return

8.2 Checking for ESDIS Standards Compliance in Fortran 90

This procedure describes how to use the Fortran 90 compiler flags on the SPR SGI machines to check science software written in Fortran 90 for ESDIS standards compliance.

Unlike with FORTRAN 77, no COTS tool is used to check Fortran 90 science software. Instead, this procedure describes how to use the compiler to perform the checking (ESDIS standards for Fortran 90 are ANSI). Since the Fortran 90 compiler is used, the checking for standards compliance can be naturally tied in with building the science software (since this procedure will produce object files suitable for linking). However, in this procedure, the building of the software (compiling *and* linking) is deferred to a later procedure.

The Activity Checklist table that follows provides an overview of the process for checking ESDIS standards compliance in Fortran 90. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where

details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.2-1 Checking for ESDIS Standards Compliance in Fortran 90 - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into the SPR SGI.	(I) 8.2	
2	SSI&T	Set up the SDP Toolkit environment.	(I) 8.2	
3	SSI&T	Set the ClearCase view.	(I) 8.2	
4	SSI&T	Go to the directory containing the source files.	(I) 8.2	
5	SSI&T	Run the Fortran 90 compiler with ANSI checking and save the results to a file.	(I) 8.2	
6	SSI&T	Examine the Fortran 90 compiler output file.	(I) 8.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Fortran 90 science software source code is available, accessible, and has read permissions for the user.
2. Required Status Message Facility (SMF) files have been compiled (see Section 9.3).
3. The C shell (or a derivative) is the current command shell.
4. The Fortran 90 compiler is available on the SPR SGI.

To check for ESDIS standards compliance in Fortran 90 code, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.

- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.

2 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname***, press **Return**. Then type, **source $PGSHOME/bin/*sgiX*/pgs-dev-env.csh**, press **Return**.

- The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version (see Section 9.2).
- The ***sgiX*** refers to the appropriate processor (see Section 9.2). For example, type **source $PGSHOME/bin/sgi/pgs-dev-env.csh**, press **Return**.

3 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of a view allowing the Fortran 90 source files to be accessible.
- This step is only necessary if any of the Fortran 90 source files are in ClearCase (in the VOB under configuration management).

4 At the UNIX prompt on the SPR SGI, type **cd *SrcPathname***, press **Return**.

- The ***SrcPathname*** is the full path name to the location of the Fortran 90 source files to be checked.
- The ***SrcPathname*** will be in the ClearCase VOB is the Fortran 90 source files are checked into ClearCase.

5 At the UNIX prompt on the SPR SGI, type **f90 -c -ansi *[*-I$PGSINC*]* *[*-I$HDFINC*]* *[[*-IOtherIncFiles*]…]* *SourceFiles* >& *ReportFile***, press **Return**.

- The terms in square brackets (*[ ]*) are used to optionally specify locations of include and module (.mod) files. The **$PGSINC** already contains the SDP Toolkit include directory and **$HDFINC** already contains the HDF include directory. The **OtherIncFiles** represents one or more additional include or module directories.
- The ***SourceFiles*** is a list (space delimited) of Fortran 90 source files or a wildcard template (*e.g.* *.f90).
- The **>&** is a C shell construct that causes standard error (where the output from the Fortran 90 compiler normally emerges) to be redirected to a file.
- The ***ReportFile*** is the file name under which to save the results of the compile process.
- The **-c** flag causes only compilation (no linking).
- The **-ansi** flag enables ANSI checking.
- Apply the terms in square brackets only as necessary. Do not include the brackets in the actual command. See example below.
- Do not use the **-I** option for include or module files that are in the standard directories or in the current directory.
- The makefile for the science software may contain the names of additional include files needed by the software.
- For example, type **f90 -c -I$PGSINC -I$HDFINC -I/ecs/modis/pge5/include/ *.f90 >& pge10.report**, press **Return**.

6 At the UNIX prompt on the SPR SGI, type **vi *ReportFile***, press **Return**.

- The ***ReportFile*** is the file name for the compilation results as produced in step 5.
- Any text editor may be used for this procedure step.

Table 8.2-2 Checking for ESDIS Standards Compliance in Fortran 90 - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to SPR SGI
2	setenv PGSHOME *ToolkitPathname* source $PGSHOME/bin/sgi*X*/pgs-dev-env.csh	press Return press Return

Table 8.2-2 Checking for ESDIS Standards Compliance in Fortran 90 - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
3	(Optional) cleartool setview *ViewName*	press Return
4	cd *SrcPathname*	press Return
5	f90 -c -ansi *[*-I$PGSINC*] [*-I$HDFINC*] [[*-IOtherIncFiles*]…]* *SourceFiles* >& *ReportFile*	press Return
6	vi *ReportFile*	press Return

8.3 Checking for ESDIS Standards Compliance in C

This procedure describes how to use the C compiler flags on the SPR SGI machines to check science software written in C for ESDIS standards compliance.

Unlike with FORTRAN 77, no COTS tool is used to check C science software. Instead, this procedure describes how to use the compiler to perform the checking (ESDIS standards for C are essentially ANSI). Since the C compiler is used, the checking for standards compliance can be naturally tied in with building the science software (since this procedure will produce object files suitable for linking). However, in this procedure, the building of the software (compiling *and* linking) is deferred to a later procedure.

The Activity Checklist table that follows provides an overview of the process for checking ESDIS standards compliance in C. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.3-1 Checking for ESDIS Standards Compliance in C - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into the SPR SGI.	(I) 8.3	
2	SSI&T	Set up the SDP Toolkit environment.	(I) 8.3	
3	SSI&T	Set the ClearCase view.	(I) 8.3	
4	SSI&T	Go to the directory containing the source files.	(I) 8.3	
5	SSI&T	Run the C compiler with ANSI checking and save the results to a file.	(I) 8.3	
6	SSI&T	Examine the C compiler output file.	(I) 8.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C science software source code is available, accessible, and has read permissions for the user.
2. Required Status Message Facility (SMF) files have been compiled (see Section 9.3).
3. The C shell (or a derivative) is the current command shell.

The C compiler is available on the SPR SGI.

To check for ESDIS standards compliance in C code, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.

- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.

2 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname***, press **Return**. Then type, source **$PGSHOME/bin/*sgiX*/pgs-dev-env.csh**, press **Return**.

- The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version (see Section 9.2).
- The ***sgiX*** refers to the appropriate processor (see Section 9.2). For example, type **source $PGSHOME/bin/sgi/pgs-dev-env.csh**, press **Return**.

3 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of a view allowing the C source files to be accessible.
- This step is only necessary if any of the C source files are in ClearCase (in the VOB under configuration management).

4 At the UNIX prompt on the SPR SGI, type **cd *SrcPathname***, press **Return**.

- The ***SrcPathname*** is the full path name to the location of the C source files to be checked.
- The ***SrcPathname*** will be in the ClearCase VOB is the C source files are checked into ClearCase.

5 At the UNIX prompt on the SPR SGI, type **cc -c -ansiposix *[*-I$PGSINC*]* *[*-I$HDFINC*]* *[[*-IOtherIncFiles*]…]* *SourceFiles* >& *ReportFile***, press **Return**.

- The terms in square brackets (*[ ]*) are used to optionally specify locations of include and module (.mod) files. The **$PGSINC** already contains the SDP Toolkit include directory and **$HDFINC** already contains the HDF include directory. The **OtherIncFiles** represents one or more additional include directories.
- The ***SourceFiles*** is a list (space delimited) of C source files or a wildcard template (*e.g.* *.c).

- The **>&** is a C shell construct that causes standard error (where the output from the C compiler normally emerges) to be redirected to a file.
- The ***ReportFile*** is the file name under which to save the results of the compile process.
- The **-c** flag causes only compilation (no linking).
- The **-ansiposix** flag enables ANSI and POSIX checking.
- Apply the terms in square brackets only as necessary. Do not include the brackets in the actual command. See example below.
- Do not use the **-I** option for include files that are in the standard directories (*e.g.* /usr/include) or in the current directory.
- The makefile for the science software may contain the names of additional include files needed by the software.
- For example, type **cc -c -I$PGSINC -I$HDFINC -I/ecs/modis/pge5/include/ *.c >& pge10.report**, press **Return**.

6 At the UNIX prompt on the SPR SGI, type **vi *ReportFile***, press **Return**.

- The ***ReportFile*** is the file name for the compilation results as produced in step 5.
- Any text editor may be used for this procedure step.

Table 8.3-2 Checking for ESDIS Standards Compliance in C - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to SPR SGI
2	setenv PGSHOME *ToolkitPathname* source $PGSHOME/bin/sgi*X*/pgs-dev-env.csh	press Return press Return
3	(Optional) cleartool setview *ViewName*	press Return
4	cd *SrcPathname*	press Return
5	cc -c -ansiposix *[*-I$PGSINC*] [*-I$HDFINC*]* *[[*-IOtherIncFiles*]…] SourceFiles* >& *ReportFile*	press Return
6	vi *ReportFile*	press Return

8.4 Checking for ESDIS Standards Compliance in Ada

This procedures describes how to use Ada compilers on the SPR SGI machines to check science software written in Ada for ESDIS standards compliance.

Unlike with FORTRAN 77, Fortran 90, or C, Ada compilers are subjected to a validation process by the DoD Ada Committee. Thus, any code that compiles successfully by a validated compiler is, by definition, fully ANSI compliant. Since the Ada compiler is used, the checking for standards compliance can be naturally tied in with building the science software (since this procedure will produce object files suitable for linking). However, in this procedure, the building of the software (compiling *and* linking) is deferred to a later procedure.

Section 8.4.1 describes how to use the COTS Verdix compiler and 8.4.2 describes how to use the GNU *gcc* compiler.

8.4.1 Checking for ESDIS Standards Compliance in Ada: Verdix COTS

This procedure describes compiling Ada software using the COTS Verdix Ada Development System (VADS) which provides a complete environment for building (and developing) Ada software. See Section 8.4.2 for use of the *gcc* compiler in compiling Ada code.

The Activity Checklist table that follows provides an overview of the process for checking for ESDIS standards compliance in Ada using the Verdix COTS. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.4.1-1 Checking for ESDIS Standards Compliance in Ada: Verdix COTS - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into the SPR SGI.	(I) 8.4.1	
2	SSI&T	Set the ClearCase view.	(I) 8.4.1	
3	SSI&T	Set the SGI_ABI environment variable.	(I) 8.4.1	
4	SSI&T	Go to the directory containing the source files.	(I) 8.4.1	
5	SSI&T	Create a VADS library directory.	(I) 8.4.1	
6	SSI&T	Run the VADS make utility.	(I) 8.4.1	
7	SSI&T	Examine the VADS output file.	(I) 8.4.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Ada science software source code is available, accessible, and has read permissions for the user.
2. The C shell (or a derivative) is the current command shell.

The Ada compiler is available on the SPR SGI.

To check for ESDIS standards compliance in Ada code, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.

- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.

2 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName***, press **Return**.
- The ***ViewName*** is the name of a view allowing the Ada source files to be accessible.
- This step is only necessary if any of the Ada source files are in ClearCase (in the VOB under configuration management).

3 At the UNIX prompt on the SPR SGI, type **setenv SGI_ABI -32**, press **Return**.
- This command sets the environment variable **SGI_ABI** for 32-bit mode compilation.

4 At the UNIX prompt on the SPR SGI, type **cd *SrcPathname***, press **Return**.
- The ***SrcPathname*** is the full path name to the location of the Ada source files to be checked.
- The ***SrcPathname*** will be in the ClearCase VOB if the Ada source files are checked into ClearCase.

5 At the UNIX prompt on the SPR SGI, type **a.mklib**, press **Return**.
- This command creates a VADS library directory. All Ada compilation must occur in a VADS Ada library.

6 At the UNIX prompt on the SPR SGI, type **a.make -v -f *SourceFiles* >& *ReportFile***, press **Return**.
- The ***SourceFiles*** is a list (space delimited) of Ada source files or a wildcard template (*e.g.* *.ada).
- The **>&** is a C shell construct that causes standard error (where the output from the Ada compiler normally emerges) to be redirected to a file.
- The ***ReportFile*** is the file name under which to save the results of the compile process.
- The **-v** flag enables verbose output.
- The **-f** flag indicates that what immediately follows are the source files. The order of the flags is therefore important.

7 At the UNIX prompt on the AIT Sun, type **vi *ReportFile***, press **Return**.
- The ***ReportFile*** is the file name for the compilation results as produced in step 6.
- Any text editor may be used for this procedure step.

Table 8.4.1-2 Checking for ESDIS Standards Compliance in Ada: Verdix COTS - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to SPR SGI
2	(Optional) cleartool setview *ViewName*	press Return

Table 8.4.1-2 Checking for ESDIS Standards Compliance in Ada: Verdix COTS - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
3	setenv SGI_ABI -32	press Return
4	cd *SrcPathname*	press Return
5	a.mklib	press Return
6	a.make -v -f *SourceFiles* >& *ReportFile*	press Return
7	vi *ReportFile*	press Return

8.4.2 Checking for ESDIS Standards Compliance in Ada: GNU *gcc* Compiler

This procedure describes compiling Ada software using the GNU C compiler, *gcc*. See Section 8.4.1 for use of the COTS Verdix compiler in compiling Ada code.

The Activity Checklist table that follows provides an overview of the process for checking ESDIS standards compliance in Ada using the *gcc* compiler. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.4.2-1 Checking for ESDIS Standards Compliance in Ada: GNU gcc Compiler - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into the SPR SGI.	(I) 8.4.2	
2	SSI&T	Set the ClearCase view.	(I) 8.4.2	
3	SSI&T	Set the SGI_ABI environment variable.	(I) 8.4.2	
4	SSI&T	Go to the directory containing the source files.	(I) 8.4.2	
5	SSI&T	Run the gcc compiler.	(I) 8.4.2	
6	SSI&T	Examine the gcc compiler output file.	(I) 8.4.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Ada science software source code is available, accessible, and has read permissions for the user.
2. The C shell (or a derivative) is the current command shell.

The GNU *gcc* compiler is available on the SPR SGI.

To check for ESDIS standards compliance in Ada code, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.

- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.
- It is recommended that this procedure begin within a new command shell on the SPR SGI.

2 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of a view allowing the Ada source files to be accessible.
- This step is only necessary if any of the Ada source files are in ClearCase (in the VOB under configuration management).

3 At the UNIX prompt on the SPR SGI, type **setenv SGI_ABI -32**, press **Return**.

- This command sets the environment variable **SGI_ABI** for 32-bit mode compilation.

4 At the UNIX prompt on the SPR SGI, type **cd *SrcPathname***, press **Return**.

- The ***SrcPathname*** is the full path name to the location of the Ada source files to be checked.
- The ***SrcPathname*** will be in the ClearCase VOB is the Ada source files are checked into ClearCase.

5 At the UNIX prompt on the SPR SGI, type **gcc -c -gnat83 *SourceFiles* >& *ReportFile***, press **Return**.

- The ***SourceFiles*** is a list (space delimited) of Ada source files or a wildcard template (*e.g.* *.ada).
- The **>&** is a C shell construct that causes standard error (where the output from the *gcc* compiler normally emerges) to be redirected to a file.
- The ***ReportFile*** is the file name under which to save the results of the compile process.
- The **-c** flag causes only compilation (no linking).
- The **-gnat83** enables compilation of Ada using the 1983 Ada Standard. Note that without this flag, the compiler would assume the 1995 Ada proposed Standard.

6 At the UNIX prompt on the SPR SGI, type **vi *ReportFile***, press **Return**.

- The ***ReportFile*** is the file name for the compilation results as produced in step 5.
- Any text editor may be used for this procedure step

Table 8.4.2-2 Checking for ESDIS Standards Compliance in Ada: GNU gcc Compiler - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to SPR SGI
2	(Optional) cleartool setview *ViewName*	press Return
3	setenv SGI_ABI -32	press Return
4	cd *SrcPathname*	press Return
5	gcc -c -gnat83 *SourceFiles* >& *ReportFile*	press Return
6	vi *ReportFile*	press Return

8.5 Checking for Prohibited Functions

The Project standards and guidelines are contained in the document *Data Production Software and Science Computing Facility (SCF) Standards and Guidelines, Revision A, October 1996* (423-16-01). This ESDIS document mandates that science software delivered to the DAACs to be integrated into the ECS be free of prohibited functions. The procedures that follow describe how to use the Prohibited Function Checker, available with a GUI and without.

Section 8.5.1 describes how to use the Prohibited Function Checker GUI. Section 8.5.2 describes how to use the Prohibited Function Checker from the command line. Both versions of the checker are functionally identical.

8.5.1 Checking for Prohibited Functions: GUI Version

This procedure describes using the GUI version of the Prohibited Function Checker to check science software for the prohibited functions.

The Activity Checklist table that follows provides an overview of the process for checking for prohibited functions using the GUI. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.5.1-1 Checking for Prohibited Functions: GUI Version - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the Prohibited Function Checker GUI.	(I) 8.5.1	
2	SSI&T	Select the analyze option.	(I) 8.5.1	
3	SSI&T	Select the source file directory.	(I) 8.5.1	
4	SSI&T	Select the source files to check.	(I) 8.5.1	

Table 8.5.1-1 Checking for Prohibited Functions: GUI Version - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
5	SSI&T	Check the files.	(I) 8.5.1	
6	SSI&T	Examine the results.	(I) 8.5.1	
7	SSI&T	Optionally, save or print the results.	(I) 8.5.1	
8	SSI&T	Optionally, view the prohibited functions.	(I) 8.5.1	
9	SSI&T	Quit the Prohibited Function Checker.	(I) 8.5.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager is running and that the source files to be checked are available, accessible, and have read permissions for the operator.
2. Source files to be checked are Ada, C, FORTRAN 77, Fortran 90, C shell, Korn shell, Bourne shell, or Perl and have recognizable file name extensions (refer to Table 8.5.1-2).

Table 8.5.1-2 File Name Extensions Recognized

Language	File Name Extensions
Ada	.a, .ada
C	.c, .h
FORTRAN 77	.f, .f77, .ftn
Fortran 90	.f90
C Shell	.csh
Korn Shell	.ksh
Bourne Shell	.sh
Perl	.pl

To check for prohibited functions in delivered source files, execute the procedure steps that follow:

Assumptions:

1. The SSIT Manager is running.

1 From the SSIT Manager, click on the **Tools** menu, then choose **Standards Checkers**. Then choose **Prohibited Function Checker**.

- The Prohibited Function Checker GUI will be displayed.

2 In the Prohibited Function Checker GUI, click on the **Analyze** button.

- The File Selector GUI will be displayed.

3 Within the **Directories** subwindow, double click on the desired directory.

- Repeat this step until the directory with the source files to be checked are displayed in the **Files** subwindow.

4 Within the **Files** subwindow, click on the source files to be checked. Each file clicked on will be highlighted.

- To choose groups of contiguous files, hold down the left mouse button and drag the mouse.
- To choose non-contiguous files, hold down the Control key while clicking on file names.

5 In the File Selector GUI, click on the **Ok** button.

- The File Selector GUI will disappear.
- The files selected in step 5 will be displayed in the Prohibited Function Checker GUI window as they are being checked.

6 In the Prohibited Function Checker GUI, click on the **Report** button.

- The **Report** GUI will be displayed.
- For each file, a list of prohibited functions found will be displayed.

7 Optionally, click on the **Print** button or the **Save** button.

- Choose **Save** to save the results to a file; choose **Print** to have the results printed on the default printer.
- Choosing **Save** will bring up a GUI labeled **Save To File**. Specify the directory and file name in which to save the results file.

8 Optionally, in the Prohibited Function Checker GUI, highlight one of the source files listed. Then click on **View**.

- The **Source Code** GUI will be displayed.
- Occurrences of prohibited functions found in that source file will be highlighted.
- Click on the **Next** button to bring into the window successive occurrences of prohibited functions (the **Next** button does not bring in the next source file).
- Click on the **Done** button to close the **Source Code** GUI. Other source files may be examined similarly, one at a time.

9 In the Prohibited Function Checker GUI, click on the **Quit** button.

- The Prohibited Function Checker GUI will disappear.
- This ends the session.

Table 8.5.1-3 Checking for Prohibited Functions: GUI Version - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Standards Checkers → Prohibited Function Checker	(No action)
2	Analyze button	click button
3	directory	click name
4	file(s)	click name(s)
5	Ok	click button
6	(Optional) Report	click button
7	(Optional) Save | Print	click button
8	(Optional) View	click button

8.5.2 Checking for Prohibited Functions: Command-Line Version

This procedure describes using the command-line version of the Prohibited Function Checker to check science software for the prohibited functions.

The Activity Checklist table that follows provides an overview of the process for checking for prohibited functions using the command-line tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.5.2-1 Checking for Prohibited Functions: Command-Line Version - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view.	(I) 8.5.2	
2	SSI&T	Go to the directory containing the source code files.	(I) 8.5.2	
3	SSI&T	Invoke the Prohibited Function Checker saving the results to an output file.	(I) 8.5.2	
4	SSI&T	Examine the output results file.	(I) 8.5.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager and its various tools have been installed in the standard directories.
2. The source files to be checked are available, accessible, and have read permissions for the operator.
3. Source files to be checked are Ada, C, FORTRAN 77, Fortran 90, C shell, Korn shell, Bourne shell, or Perl and have recognizable file name extensions (refer to Table 8.5.1-2).

To check for prohibited functions in delivered source files, execute the procedure steps that follow:

1 If required, at the UNIX prompt on an AIT Sun, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of a view allowing the source files to be accessible.
- This step is only necessary if any of the source files are in ClearCase (in the VOB under configuration management).

2 At the UNIX prompt on the AIT Sun, type **cd *SrcPathname***, press **Return**.

- The ***SrcPathname*** is the full path name to the location of the source files to be checked.
- The ***SrcPathname*** will be in the ClearCase VOB if the source files are checked into ClearCase.
- The ***SrcPathname*** can contain other directories that contain source files and/or more directories. The Prohibited Function Checker will search out all source files in subdirectories recursively.

3 At the UNIX prompt on the AIT Sun, type **$DPATMGR_HOME/data/DPS/DpAtMgrBadFunc ConfigFile /ecs/formal/Rel_A/CUSTOM/cfg/DpAtBA_*daac*.CFG *FilesOrDirectories* > *ResultsFile***, press **Return**.

- The ***daac*** is one of { GSFC, EDC, LARC, NSIDC}.
- The ***FilesOrDirectories*** is a list of source file names or directory names of directories containing source files.
- The ***ResultsFile*** is the file name for the results that are output.
- For example, type **$DPATMGR_HOME/data/DPS/DpAtMgrBadFunc ConfigFile /ecs/formal/Rel_A/CUSTOM/cfg/DpAtBA_NSIDC.CFG main.c utils/ > myOutput**, press **Return**. Here, **main.c** is a source file and **utils/** is a directory that contains other source files.

4 At the UNIX prompt on the AIT Sun, type **vi *ResultsFile***, press **Return**.

- The ***ResultsFile*** is the file name for the output results as produced in step 3.
- Any text editor may be used for this procedure step.

Table 8.5.2-2 Checking for Prohibited Functions: Command-Line Version - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cleartool setview *ViewName*	press Return
2	cd *SrcPathname*	press Return
3	$DPATMGR_HOME/data/DPS/DpAtMgrBadFunc ConfigFile /ecs/formal/Rel_A/CUSTOM/cfg/DpAtBA_*daac*.CFG *FilesOrDirectories* > *ResultsFile*	press Return
4	vi *ResultsFile*	press Return

8.6 Checking Process Control Files

Process Control Files (PCFs) should be delivered for each Product Generation Executive (PGE). Only one PCF can be associated with a PGE, however, more than one may be delivered. This procedure describes how to check PCFs for valid syntax and format using the Process Control File Checker, available with a GUI and without.

Section 8.6.1 describes how to use the Process Control File Checker GUI. Section 8.6.2 describes how to use the Process Control File Checker from the command line. Both versions of the checker are functionally identical.

8.6.1 Checking Process Control Files: GUI Version

This procedure describes using the GUI version of the Process Control File Checker to check process control files delivered with the science software.

The Activity Checklist table that follows provides an overview of the process for checking Process Control Files using the GUI. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.6.1-1 Checking Process Control Files: GUI Version - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the Process Control File Checker GUI.	(I) 8.6.1	
2	SSI&T	Select the PCF directory.	(I) 8.6.1	
3	SSI&T	Select the PCF to check.	(I) 8.6.1	
4	SSI&T	Check the PCF.	(I) 8.6.1	

Table 8.6.1-1 Checking Process Control Files: GUI Version - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
5	SSI&T	Optionally, save or print the results.	(I) 8.6.1	
6	SSI&T	Select another PCF or quit.	(I) 8.6.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager is running.
2. The PCFs to be checked are available, accessible, and have read permissions for the operator.

To check Process Control Files, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Standards Checkers**. Then choose **Process Control File Checker**.
- The Process Control File Checker GUI will be displayed.

2 In the **Directories** subwindow, double click on the desired directory.
- Repeat this step until the directory with the PCF(s) to be checked is displayed in the Files window.
- Use the **Filter** subwindow to limit which files are displayed.

3 Within the **Files** subwindow, click on the PCF to be checked**.**
- The file clicked on will be highlighted.
- Only one PCF can be checked at a time.

4 Click on the **Check PCF** button.
- A GUI labeled **PCF Checker Results** will be displayed.
- Results will be displayed in this window.

5 Optionally, click on the **Save** button or on the **Print** button.
- Choose **Save** to save the results to a file; choose **Print** to have the results printed on the default printer.
- Choosing **Save** will bring up a GUI labeled **Save To File**. Specify the directory and file name in which to save the results file.
- Choosing **Print** and then clicking on the **OK** button will send the results to the default printer.

6 Click on the **Check Another** button or on the **Quit** button.
- Choosing **Check Another** allows another PCF to be checked. Repeat steps 2 through 5.

- Choosing **Quit** causes the Process Control File Checker GUI to disappear and ends the session.

Table 8.6.1-2 Checking Process Control Files: GUI Version - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Standards Checkers → Process Control File Checker	(No action)
2	directory	click name
3	file(s)	click name(s)
4	Check PCF	click button
5	(Optional) Save | Print	click button
6	Check Another | Quit	click button

8.6.2 Checking Process Control Files: Command-Line Version

This procedure describes using the command-line version of the Process Control File Checker to check process control files delivered with the science software.

The Activity Checklist table that follows provides an overview of the process for checking Process Control Files using the command-line tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.6.2-1 Checking Process Control Files: Command-Line Version - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the ClearCase view.	(I) 8.6.2	
2	SSI&T	Go to the directory containing the Process Control File(s)	(I) 8.6.2	
3	SSI&T	Invoke the Process Control File Checker and save results to output file.	(I) 8.6.2	
4	SSI&T	Examine the results output file.	(I) 8.6.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PCF files to be checked are available, accessible, and have read permissions for the operator.
2. The directory, /usr/ecs/Rel_A/CUSTOM/bin/daac_toolkit_f77/TOOLKIT/bin/sun5 is in the user's path on the AIT machines. To verify, type **which pccheck.sh**, press **Return**. If a path is displayed, then the directory is in your path.

To check Process Control Files, execute the procedure steps that follow:

1 If required, at the UNIX prompt on an AIT Sun, type **cleartool setview *ViewName***, press **Return**.

- The ***ViewName*** is the name of a view allowing the Process Control File(s) to be accessible.
- This step is only necessary if any of the Process Control Files are in ClearCase (in the VOB under configuration management).

2 At the UNIX prompt on AIT Sun, type **cd *PCFpathname***, press **Return**.

- The ***PCFpathname*** is the full path name to the location of the Process Control File(s) to be checked.
- The ***PCFpathname*** will be in the ClearCase VOB if the Process Control Files are checked into ClearCase.

3 At the UNIX prompt on an AIT Sun, type **pccheck.sh -i *PCFfilename* > *ResultsFile***, press **Return**.

- The ***PCFfilename*** is the full path name (directory and file name) to the Process Control File to check.
- The ***ResultsFile*** is the file name for the results that are output.
- The PCF Checker is also available on the SPR SGI machines. The easiest way to access it is to set a SDP Toolkit environment (any will do for purposes here, see Section 9.2) and type **$PGSBIN/pccheck.sh -i *PCFfilename* > *ResultsFile***, press **Return**.

4 At the UNIX prompt on the SPR SGI, type **vi *ResultsFile***, press **Return**.

- The ***ResultsFile*** is the file name for the output results as produced in step 4.
- Any text editor may be used for this procedure step.

Table 8.6.2-2 Checking Process Control Files: Command-Line Version - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	(Optional) cleartool setview *ViewName*	press Return
2	cd *PCFpathname*	press Return

Table 8.6.2-2 Checking Process Control Files: Command-Line Version - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
3	pccheck.sh -i *PCFfilename* > *ResultsFile*	press Return
4	vi *ResultsFile*	press Return

8.7 Extracting Prologs

The Project standards and guidelines are contained in the document *Data Production Software and Science Computing Facility (SCF) Standards and Guidelines, Revision A, October 1996* (423-16-01). This ESDIS document mandates that science software delivered to the DAACs to be integrated into the ECS contain prologs in the source files. Prologs are internal documentation containing information about the software. The details are specified in the ESDIS document. Prologs must be at the top of every function, subroutine, procedure, or program module.

This procedure describes using the Prolog Extractor to extract prologs into a file. Note that the prolog extractor only extract the prologs it finds. It does not check the contents of prologs.

The Activity Checklist table that follows provides an overview of the process for extracting prologs. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 8.7-1 Extracting Prologs - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the Prolog Extractor.	(I) 8.7	
2	SSI&T	Specify the configuration file.	(I) 8.7	
3	SSI&T	List out the files and/or directories to be checked for prologs.	(I) 8.7	
4	SSI&T	Quit the Prolog Extractor	(I) 8.7	
5	SSI&T	Optionally, view the extracted prologs in a text editor.	(I) 8.7	
6	SSI&T	Quit the editor.	(I) 8.7	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager is running.
2. The source files from which prologs are to be extracted are available, accessible, and have read permissions for the operator.
3. Prologs are delimited by particular delimiters depending on the language type. Delimiters are listed in the Table 8.7-2.
4. Source files have file name extensions shown in Table 8.7-3.

Table 8.7-2 Extracting Prologs - Prolog Delimiters

Language	Type	Delimiter
FORTRAN 77	source	!F77
Fortran 90	source	!F90
C	source	!C
Ada	source	!Ada
FORTRAN 77	include	!F77-INC
Fortran 90	include	!F90-INC
C	include	!C-INC
Any Language	any	!PROLOG
All Languages	The end delimiter is always !END	

Table 8.7-3 Extracting Prologs - File Name Extensions

File Type	File Name Extensions
FORTRAN 77	f, f77, ftn, for, F, F77, FTN, FOR
Fortran 90	f90, F90, f, F
FORTRAN 77/Fortran 90 include	inc, INC
C	c
C/C++ header	h
Ada	a, ada

To extract prologs from delivered source files, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Standards Checkers**. Then choose **Prolog Extractor** .

- An xterm (on the AIT Sun) will be displayed within which the Prolog Extractor will be run.
- The Prolog Extractor can also be started from the UNIX prompt. To do this, at the UNIX prompt on the AIT Sun, type **DpAtMgrPrologs**, press **Return**.

2 At the program prompt **Config filename (default *defaultConfigFile*)?**, press **Return**.

- The default configuration file for this tool will be used.
- The ***defaultConfigFile*** will be replaced by the full path name and file name of the default configuration file. The file name will be DpAtPL_*daac*.CFG where *daac* will be replaced by one of {GSFC, EDC, LARC, NSIDC}.

3 At the **Files(s)? (-h help)** prompt, type in file names and/or directory names containing the files.

- Separate items with spaces.
- If item is a directory (and it exists within the current directory), the contents of the directory will be search recursively for files with valid file name extensions (see Table 8.7-3).
- Use **./** to indicate the current directory. The extractor will search for all files in the current directory and recursively below.
- The current directory is the directory from which the SSIT Manager was started.
- The time needed for the Prolog Extractor could be very long for large numbers of files and directories.
- When extraction is complete, the message **Output written to file: ./prologs.txt** will be displayed.

4 At the program prompt **Hit return for another, 'q <return>' to quit:**, press **Return** to repeat process with another set of source files or type **q** and press **Return** to quit.

- The xterm will disappear.

5 At a UNIX prompt on the AIT Sun, type **vi prologs.txt**, press **Return**.

- The extracted prologs file, named **prologs.txt**, will be brought into the editor.
- Any text editor may be used such as *emacs*. For example, **emacs prologs.txt**, press **Return**.
- The default location of the **prologs.txt** file is the directory from which the SSIT Manger was invoked.

6 Once the extracted prologs file has been examined, exit the editor.

- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor's documentation.

Table 8.7-4 Extracting Prologs - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Standards Checkers → Prolog Extractor	(No action)
2	(No entry)	press Return
3	list file names and/or directory names	press Return
4	q | Return	press Return
5	vi prologs.txt	press Return
6	:wq	press Return

This page intentionally left blank.

9. Compiling and Linking Science Software

Science software delivered to the DAACs for SSI&T is in the form of source files. In order to be run and tested within the ECS, this science software has to be compiled and linked to form the binary executables that run within the Product Generation Executives (PGEs). Science software is developed at independent Science Computing Facilities (SCFs) using the SDP Toolkit. The SDP Toolkit allows science software to be developed for the ECS at independent SCFs. Once delivered to the DAACs for SSI&T, the science software needs to be compiled and linked to one of the SDP Toolkit versions resident at the DAAC.

The procedures which follow describe how to build (compile and link) the science software.

9.1 Updating the Process Control Files (PCFs)

The process control files delivered to the DAACs for SSI&T were created and used at the SCFs. As such, path names specified in the PCF (and perhaps, file names) will need to be updated since they will not reflect path names in the DAAC environment. The same may be true of file names if they get changed after delivery to the DAAC. Note that these changes are only necessary when the PGE is to be run in an SCF simulated environment (using the SCF version of the SDP Toolkit and run on the command line). When run within the PDPS, the PCF used is generated automatically by the system.

The Activity Checklist table that follows provides an overview of the process for updating the Process Control Files. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 9.1-1 Updating the Process Control Files (PCFs) - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into the SPR SGI.	(I) 9.1	
2	SSI&T	If necessary, set the ClearCase view.	(I) 9.1	
3	SSI&T	Change directories to the location of the PCF.	(I) 9.1	
4	SSI&T	If necessary, check out the PCF from ClearCase.	(I) 6.6	
5	SSI&T	Run the Process Control File Checker.	(I) 8.6	
6	SSI&T	Invoke text editor with the PCF.	(I) 9.1	

Table 9.1-1 Updating the Process Control Files (PCFs) - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
7	SSI&T	Verify or modify default directory path names for each PCF section.	(I) 9.1	
8	SSI&T	Verify or modify path names for all science software specific entries.	(I) 9.1	
9	SSI&T	Verify presence of the shared memory pointer file.	(I) 9.1	
10	SSI&T	Save the file and quit the editor.	(I) 9.1	
11	SSI&T	Re-run the Process Control File Checker.	(I) 8.6	
12	SSI&T	If necessary, check the modified PCF back into ClearCase.	(I) 9.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. A PCF for the PGE has been delivered and is available, accessible, and has read permissions.

To update the PCF, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.
- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.

2 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName***, press **Return**.
- The ***ViewName*** is the name of a view allowing the PCF to be accessible.
- This step is only necessary if the PCF is in ClearCase (in the VOB under configuration management).

3 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **cd *PCFpathname***, press **Return**.
- The ***PCFpathname*** is the full path name to the location of the PCF. This location will be in the ClearCase VOB if the PCF is under configuration management.

4 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **cleartool checkout -nc *PCFfilename***, press **Return**.
- The ***PCFfilename*** is the file name of the PCF that is to be checked out (and later modified). The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

5 Run the Process Control File Checker on the delivered PCF (see Section 8.6).

- This will verify that the delivered PCF is correct before editing.

6 At a UNIX prompt on the AIT Sun, type **vi *PCFfilename***, press **Return**.
- The ***PCFfilename*** is the file name of the PCF to update.
- Any text editor may be used such as *emacs*. For example, **emacs AST02.pcf**, press **Return**.

7 In the file, make changes to the default directories specified in each section of the PCF. All path names specified in the PCF must exist on the SPR SGI.
- Each section begins with a line consisting of a **?** in the first column followed by a label:

```
? PRODUCT INPUT FILES
? PRODUCT OUTPUT FILES
? SUPPORT INPUT FILES
? SUPPORT OUTPUT FILES
? INTERMEDIATE INPUT
? INTERMEDIATE OUTPUT
? TEMPORARY I/O
```

- Each of the above section heading lines will then be followed (not necessarily immediately; there may be comment lines) by a line that begins with a **!** in the first column. These lines specify the default path names for each section.
  - If the line reads:

    ```
    ! ~/runtime
    ```

    leave it unchanged. The tilde (~) is a symbol that represents $PGSHOME.
  - If another path name is listed instead, it will probably need to be changed to a path name that exists at the DAAC on the SPR SGI. When specifying a path name, use an absolute path name, not a relative path name.

8 In the file, look for science software specific entries in each section and make changes to the path names (field 3) as necessary. All path names specified in the PCF must exist on the SPR SGI.
- The science software specific entries will have logical IDs (first field) *outside* of the range 10,000 to 10,999.
- Where necessary, replace the path names in the third field of each entry with the path names appropriate to the DAAC environment.
- Do not alter file entries that are used by the SDP Toolkit itself. These have logical IDs *in* the range 10,000 to 10,999.
- For example, if the following entry was found in the PCF:

  ```
  100|L1A.granule|/MODIS/run/input||||1
  ```

  change /MODIS/run/input to the appropriate path name in the DAAC where the file L1A.granule is stored.
- When specifying a path name, use an absolute path name, not a relative path name.
- Do not include the file name with the path name. The file name belongs in field 2 by itself.

9 In the file, verify that the SUPPORT OUTPUT FILES section contains an entry to the shared memory pointer file.

- Look for the entry:

```
10111|ShmMem|~/runtime||||1
```

  The third field may be blank; this will work too.
- If this entry is not within this section, add it.

10 Once changes have been made to the PCF, save the changes and exit the editor.
- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor's documentation.

11 Again, run the Process Control File Checker on the PCF (see Section 8.6) to make sure that no errors were introduced during editing.

12 If the PCF had been checked out of ClearCase, at the UNIX prompt on the SPR SGI, type **cleartool checkin -nc *PCFfilename***, press **Return**.
- The ***PCFfilename*** is the file name of the modified PCF. The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the check in step.

Table 9.1-2 Updating the Process Control Files (PCFs) - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to SPR SGI
2	(If necessary) cleartool setview *ViewName*	press Return
3	cd *PCFpathname*	press Return
4	cleartool checkout -nc *PCFfilename*	press Return
5	(No entry)	run Process Control File Checker
6	vi *PCFfilename*	press Return
7	(No entry)	modify (if necessary) default directory path names in each PCF section
8	(No entry)	modify (if necessary) path names of all science software specific entries
9	(No entry)	verify presence of shared memory pointer file
10	:wq	press Return
11	(No entry)	re-run Process Control File Checker
12	(If necessary) cleartool checkin -nc *PCFfilename*	press Return

9.2 Setting Up the SDP Toolkit Environment

Science software developed and tested at the SCFs are built using the SCF version of the SDP Toolkit. This version of the SDP Toolkit enables the SCFs to simulate to some extent a DAAC environment. The DAAC version of the SDP Toolkit uses an identical Applications

Programming Interface (API) as the SCF version, however, it contains additional hooks into the ECS and provide full ECS functionality. Science software developed at the SCFs with the SCF version of the SDP Toolkit should be able to build and run with the DAAC version of the SDP Toolkit with no changes.

Each SDP Toolkit version (SCF and DAAC) has different flavors depending upon the object type mode and upon the language. The SGI Power Challenge (the SPR machines) run with the IRIX 6.2 operating system. This operating system allows compilers to build binaries in old 32-bit mode, new 32-bit mode, or in 64-bit mode (default). Table 9.2-1 lists the modes available and the corresponding C compiler flags to enable them.

Table 9.2-1 Object Types on the SGI

Object Type	C Compiler Flag Used
Old 32-bit Mode	-32
New 32-bit Mode	-n32
64-bit Mode	-64

In addition, the SDP Toolkit uses different libraries depending upon whether FORTRAN 77 or Fortran 90 source code is being linked. If only C source code is being linked, then either language library can be used.

Given that there are three object types (see Table 9.2-1) and each has two language types (see above paragraph), there are six versions of the DAAC Toolkit at the DAACs. Since the SCF version of the Toolkit is also available, a total of 12 Toolkit versions is available. Table 9.2-2 lists these versions. This table lists the Toolkit version (SCF or DAAC), language type, library object mode, and the home directory of the appropriate SDP Toolkit (represented by the environment variable PGSHOME).

Table 9.2-2 SDP Toolkit Versions at the DAAC

SDP Version	Language Type	Library Object Type	$PGSHOME	$PGSBIN
SCF	FORTRAN 77 or C	Old 32-bit mode	$CUSTOM_HOME/bin/ scf_toolkit_f77/TOOLKIT/	$CUSTOM_HOME/bin/ scf_toolkit_f77/TOOLKIT/ bin/sgi/
SCF	Fortran 90 or C	Old 32-bit mode	$CUSTOM_HOME/bin/ scf_toolkit_f90/TOOLKIT/	$CUSTOM_HOME/bin/ scf_toolkit_f90/TOOLKIT/ bin/sgi/
SCF	FORTRAN 77 or C	New 32-bit mode	$CUSTOM_HOME/bin/ scf_toolkit_f77/TOOLKIT/	$CUSTOM_HOME/bin/ scf_toolkit_f77/TOOLKIT/ bin/sgi32/

SCF	Fortran 90 or C	New 32-bit mode	$CUSTOM_HOME/bin/ scf_toolkit_f90/TOOLKIT/	$CUSTOM_HOME/bin/ scf_toolkit_f90/TOOLKIT/ bin/sgi32/
SCF	FORTRAN 77 or C	64-bit mode	$CUSTOM_HOME/bin/ scf_toolkit_f77/TOOLKIT/	$CUSTOM_HOME/bin/ scf_toolkit_f77/TOOLKIT/ bin/sgi64/
SCF	Fortran 90 or C	64-bit mode	$CUSTOM_HOME/bin/ scf_toolkit_f90/TOOLKIT/	$CUSTOM_HOME/bin/ scf_toolkit_f90/TOOLKIT/ bin/sgi6432/
DAAC	FORTRAN 77 or C	Old 32-bit mode	$CUSTOM_HOME/bin/ daac_toolkit_f77/TOOLKIT/	$CUSTOM_HOME/bin/ daac_toolkit_f77/TOOLKIT/ bin/sgi/
DAAC	Fortran 90 or C	Old 32-bit mode	$CUSTOM_HOME/bin/ daac_toolkit_f90/TOOLKIT/	$CUSTOM_HOME/bin/ daac_toolkit_f90/TOOLKIT/ bin/sgi/
DAAC	FORTRAN 77 or C	New 32-bit mode	$CUSTOM_HOME/bin/ daac_toolkit_f77/TOOLKIT/	$CUSTOM_HOME/bin/ daac_toolkit_f77/TOOLKIT/ bin/sgi32/
DAAC	Fortran 90 or C	New 32-bit mode	$CUSTOM_HOME/bin/ daac_toolkit_f90/TOOLKIT/	$CUSTOM_HOME/bin/ daac_toolkit_f90/TOOLKIT/ bin/sgi32/
DAAC	FORTRAN 77 or C	64-bit mode	$CUSTOM_HOME/bin/ daac_toolkit_f77/TOOLKIT/	$CUSTOM_HOME/bin/ daac_toolkit_f77/TOOLKIT/ bin/sgi64/
DAAC	Fortran 90 or C	64-bit mode	$CUSTOM_HOME/bin/ daac_toolkit_f90/TOOLKIT/	$CUSTOM_HOME/bin/ daac_toolkit_f90/TOOLKIT/ bin/sgi64/

$CUSTOM_HOME is an environment variable set to /usr/ecs/Rel_A/CUSTOM

The choice of which version of the SDP Toolkit to use depends upon two factors: the test being performed and the version is required by the science software. For running a PGE in a simulated SCF environment (*i.e.* as if at the SCF), a SCF version of the Toolkit should be used (see Section 9). For running a PGE in the fully functional DAAC environment, the DAAC version should be used.

Among the DAAC versions, there are still six choices. Most science software will likely require one of the 32-bit versions. If the science software is written in C only, then the language type does not matter and either language version of the Toolkit may be used. If, however, FORTRAN 77 code is being used (with or without C), then the FORTRAN 77 language version of the DAAC Toolkit must be used. Conversely, if Fortran 90 code is being used (again, with or without C), the Fortran 90 language version of the DAAC Toolkit must be used.

If both FORTRAN 77 and Fortran 90 are being used, the procedure becomes more complex. Under such circumstances, refer to the *Release A SCF Toolkit User's Guide* for details.

This procedure describes how to set up the appropriate SDP Toolkit environment. It involves two basic steps. First, set the SDP Toolkit home directory in the environment variable PGSHOME. The second step is to source (run) the set up script in the appropriate bin directory. This step results in a number of other environment variables getting set that will be needed.

The Activity Checklist table that follows provides an overview of the process setting up the SDP Toolkit environment. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 9.2-3 Setting Up the SDP Toolkit Environment - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Set the PGSHOME environment variable.	(I) 9.2	
2	SSI&T	Run the script to set other environment variables.	(I) 9.2	
3	SSI&T	Optionally, create an alias to set environment.	(I) 9.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative) is the current command shell.

To check set up a SDP Toolkit environment, execute the procedure steps that follow:

1 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname***, press **Return**.
- The ***ToolkitPathname*** is the home directory of the particular SDP Toolkit version being used. Refer to Table 9.2-2. Note that the setting of PGSHOME shown in this table may differ in your local DAAC.
- Korn shell users, type **PGSHOME=*ToolkitPathname*; export PGSHOME**, press **Return**.

2 At the UNIX prompt on the SPR SGI, type **source $PGSHOME/bin/*sgiX*/pgs-dev-env.csh**, press **Return**.
- The ***sgiX*** is one of: **sgi** for 32-bit version of the Toolkit or **sgi64** for 64-bit version of the Toolkit. Refer to the last column of Table 9.2-2 for path names to the file to source.
- Korn shell users, type **. $PGSHOME/bin/*sgiX*/pgs-dev-env.ksh**, press **Return** (note the “dot” and then space at the beginning of this command).

3 This step is optional. Edit the file $HOME/.cshrc and add the line **alias *aliasname* 'setenv PGSHOME *ToolkitPathname*; source $PGSHOME/bin/*sgiX*/pgs-dev-env.csh; echo "*textmessage*" '**.

- The ***aliasname*** is the name of the alias. For example, to set up an environment for the DAAC version of the Toolkit for FORTRAN 77 (or C), you might use **DAACf77** as an ***aliasname***.
- The ***ToolkitPathname*** is the home directory of the particular SDP Toolkit version being used. Refer to Table 9.2-2. Note that the setting of PGSHOME shown in this table may differ in your local DAAC.
- The ***sgiX*** is one of: **sgi** for 32-bit version of the Toolkit or **sgi64** for 64-bit version of the Toolkit.
- The ***textmessage*** is a message that will be echoed to the screen signifying that a new Toolkit environment has been set up. It must be enclosed within double quotes ("). An example may be, **"DAAC F77 Toolkit environment is now set."**
- A complete example (it should be all on one line in the .cshrc file):

  alias DAACf77 'setenv PGSHOME /RelA/daac_toolkit_f77/TOOLKIT; source $PGSHOME/bin/sgi/pgs-dev-env.csh; echo "DAAC F77 Toolkit environment is now set" '
- Other aliases for other versions of the Toolkit can be set up similarly.

Table 9.2-4 Setting Up the SDP Toolkit Environment - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	setenv PGSHOME *ToolkitPathname*	press Return
2	source $PGSHOME/bin/*sgiX*/pgs-dev-env.csh	press Return
3	(Optional) alias *aliasname* 'setenv PGSHOME *ToolkitPathname*; source $PGSHOME/bin/*sgiX*/pgs-dev-env.csh; echo "*textmessage*" '	add line to .cshrc file

9.3 Compiling Status Message Facility (SMF) Files

Status Message Facility (SMF) files are used by the SDP Toolkit to facilitate a status and error message handling mechanism for use in the science software and to provide a means to send log files, informational messages, and output data files to DAAC personnel or to remote users.

Science software making use of the SMF need particular header (include) files when being built and also need particular runtime message files when being run. Both the header and message files are produced by running a SMF "compiler" on a message text file. These message text files should be part of the science software delivery to the DAAC. They typically have a .t file name extension.

This procedure describes how to compile the SMF message text files to produce both the necessary include files and the necessary runtime message files.

The Activity Checklist table that follows provides an overview of the process for compiling Status Message Facility (SMF) files. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 9.3-1 Compiling Status Message Facility (SMF) Files - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into the SPR SGI.	(I) 9.3	
2	SSI&T	Set the ClearCase view.	(I) 9.3	
3	SSI&T	Set up the SDP Toolkit environment.	(I) 9.3	
4	SSI&T	Go to the SMF directory.	(I) 9.3	
5	SSI&T	Run the SMF compiler.	(I) 9.3	
6	SSI&T	If necessary, move the created files to proper directories.	(I) 9.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative) is the current command shell.

To check compile status message facility (SMF) files, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.
- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.
- It is recommended that this procedure begin within a new command shell on the SPR SGI.

2 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName***, press **Return**.
- The ***ViewName*** is the name of a view allowing the SMF files to be accessible.
- This step is only necessary if any of the SMF files are in ClearCase (in the VOB under configuration management).

3 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname***, press **Return**. Then type, source **$PGSHOME/bin/*sgiX*/pgs-dev-env.csh**, press **Return**.
- The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version (see Section 9.2).

- The ***sgiX*** refers to the appropriate processor (see Section 9.2). For example, type **source $PGSHOME/bin/sgi/pgs-dev-env.csh**, press **Return**.

4 At the UNIX prompt on the SPR SGI, type **cd** ***pathname***, press **Return**.
- The ***pathname*** is the full path name to the directory containing the SMF text files.
- The SMF text files will typically have .t file name extensions.

5 At the UNIX prompt on the SPR SGI, type **smfcompile -*lang* -f *textfile*.t**, press **Return**.
- The **-*lang*** is a flag that indicates for what language to compile. This flag can be one of **-c** to produce C header files, **-f77** to produce FORTRAN 77 include files, and **-ada** to produce Ada include files. The default is for C include files. For example, type **smfcompile -f77 PGS_MODIS_39123.t**, press **Return**.
- The ***textfile*** is the file name of the SMF text file delivered with the science software.
- The SMF text files will typically have .t file name extensions.
- File names for SMF text files usually have the "seed" value used by the file as part of its file name (*e.g.* PGS_MODIS_39123.t where 39123 is the seed number).
- Only one such SMF text file can be compiled at a time; wildcards cannot be used.
- The SMF compiler may be run with the additional flags **-r** and **-i** as in, **smfcompile -f *textfile*.t -r -i**. The **-r** automatically places the runtime message file in the directory given by the environment variable PGSMSG. The **-i** automatically places the include file in the directory given by the environment variable PGSINC. For example, type **smfcompile -ada -r -i -f PGS_MODIS_39123.t**, press **Return**. Note that the **-f** flag must always be immediately followed by the name of the text file.

6 If necessary, at the UNIX prompt on the SPR SGI, type **mv *IncludeFilename* $PGSINC**, press **Return**. Then, type **mv *RuntimeFilename* $PGSMSG**, press **Return**.
- This step is only required if either the **-r** or the **-i** flag were not used in step 5.
- The ***IncludeFilename*** is the name of the include file created in step 5.
- The ***RuntimeFilename*** is the name of the runtime message file created in step 5.
- For example, type **mv PGS_MODIS_39123.h $PGSINC**, press **Return**. And then type, **mv PGS_MODIS_39123 $PGSMSG**, press **Return**.

Table 9.3-2 Compiling Status Message Facility (SMF) Files - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to SPR SGI
2	(Optional) cleartool setview *ViewName*	press Return
3	setenv PGSHOME *ToolkitPathname* source $PGSHOME/bin/sgi*X*/pgs-dev-env.csh	press Return press Return
4	cd *pathname*	press Return

Table 9.3-2 Compiling Status Message Facility (SMF) Files - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
5	smfcompile -*lang* -f *textfile*.t	press Return
6	(If necessary) mv *IncludeFilename* $PGSINC mv *RuntimeFilename* $PGSMSG	 press Return press Return

9.4 Building Science Software with the SCF Version of the SDP Toolkit

In order to be tested at the DAAC, science software must be compiled and linked to produce binary executables. These binary executables are then packaged into one or more shell scripts as defined by the science software developer (Instrument Team). These science software packages are the Product Generation Executives (PGEs) delivered to the DAACs during SSI&T. PGEs are the smallest schedulable unit of science software in the ECS.

Building science software into PGEs should be done in accordance with supplied documentation. Such documentation should describe the process in detail. In general, science software deliveries will come with make files or other build scripts to automate the build process.

In general, science software will be built, run, and tested with the SCF version of the SDP Toolkit to ensure that the software has been successfully ported to the DAAC. Once this test has been completed successfully, the science software will be re-built, rerun, and re-tested with the DAAC version of the SDP Toolkit. Only with the DAAC Toolkit can the PGE be run within the ECS.

This procedure describes some general principals that may or may not be applicable to a particular science software delivery for building a PGE with the SCF version of the SDP Toolkit. See Section 9.5 for building a PGE with the DAAC version of the SDP Toolkit.

The Activity Checklist table that follows provides an overview of the process for building science software with the SCF version of the SDP Toolkit. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 9.4-1 Building Science Software with the SCF Version of the SDP Toolkit - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Read all documentation supplied in the delivery.	(I) 9.4	
2	SSI&T	Log into the SPR SGI.	(I) 9.4	
3	SSI&T	Set up the SCF version of the SDP Toolkit environment.	(I) 9.4	
4	SSI&T	Set the ClearCase view.	(I) 9.4	
5	SSI&T	Examine and alter (if necessary) any supplied make file.	(I) 9.4	
6	SSI&T	Compile any Status Message Facility (SMF) files	(I) 9.3	
7	SSI&T	Verify the directory structure.	(I) 9.4	
8	SSI&T	If necessary, check out of ClearCase executable or object files.	(I) 9.4	
9	SSI&T	Perform the build.	(I) 9.4	
10	SSI&T	If necessary, check make file(s) back into ClearCase.	(I) 9.4	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative) is the current command shell.

To build science software with the SCF version of the SDP Toolkit, be aware of the “typical” procedure steps that follow:

1 Read all instructional material supplied with the science software delivery. Such material should be the primary source of information on how to build the science software.

- Read the *Systems Description* document and the *Operations Manual*. Both of these or their equivalent should be in the delivery.
- Typically, there will be “readme” files accompanying each PGE in the directory structure, perhaps in a doc directory.
- Text files (ASCII) may be viewed with the UNIX command, *more* or with the *vi* editor.
- PostScript documents may be viewed with *ghostview*, which is accessible via the SSIT Manager.
- PDF formatted documents may be viewed with *acroread*, the Acrobat Reader, also accessible via the SSIT Manager.
- Documents in Microsoft Word and related formats may be viewed through the Microsoft Windows™ 3.1 emulator. The MS Windows emulator may be accessed from the SSIT Manager.

2 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.

- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.
- It is recommended that this procedure begin within a new command shell on the SPR SGI.

3 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname***, press **Return**. Then type, source **$PGSHOME/bin/*sgiX*/pgs-dev-env.csh**, press **Return**.

- The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version, in this case, an SCF version (see Section 9.2).
- The ***sgiX*** refers to the appropriate processor (see Section 9.2). For example, type **source $PGSHOME/bin/sgi/pgs-dev-env.csh**, press **Return**.

4 If make files are in ClearCase, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName***, press **Return**. Then, **cd *pathname***, press **Return**. And **cleartool checkout -nc *makefile***, press **Return**.

- The ***ViewName*** is the name of a view allowing the make files to be accessible.
- The ***pathname*** is the full path name of the directory (in the VOB) where the make file has been checked in.
- The ***makefile*** is the name of the make file to examine and possibly modify.
- This step is only necessary if any of the make files (or build scripts) are in ClearCase (in the VOB under configuration management).

5 Examine and alter (if necessary) any make files using any text editor (*vi*, *emacs*).

- There may be several make files for a particular PGE.
- Verify that compiler, compiler flag settings, and other environment variable settings are appropriate.
- The Toolkit set up (from step 3) will set many environment variables which can be used in the make files. To see the current environment variable settings, at the UNIX prompt on the SPR SGI, type **env**, press **Return**.

6 Compile any required status message facility (SMF) files and place the header file(s) in the proper directory for building. See Section 9.3.

7 Verify that the directory structure for the PGE source files matches the directory structure expected by the make files or build scripts.

- Deliveries may come with install scripts that place files into various directories according to some predefined structure.

8 If necessary, at the UNIX prompt on the SPR SGI, type **cleartool checkout -nc *filename***, press **Return**.

- The ***filename*** is the file name of the executable, object file, or make file to be checked out of ClearCase. The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

- Note that checking in executable or object files is *not* recommended in the first place.

9 Build the software in accordance with instructions delivered.

- Science software deliveries may come with a single, top-level script to do the entire build or the build process could involve a series of steps, each of which should be described fully in the delivered documentation.

10 If necessary, at the UNIX prompt on the SPR SGI, type **cleartool checkin *filename* -nc**, press **Return**.

- The ***filename*** is the file name of the executable, object file, or make file to be checked into ClearCase. The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the check in step.
- Note that checking in executable or object files is *not* recommended.

Table 9.4-2 Building Science Software with the SCF Version of the SDP Toolkit - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	(No entry)	read all documentation
2	Tools → Xterm	telnet to SPR SGI
3	setenv PGSHOME *ToolkitPathname* source $PGSHOME/bin/sgi*X*/pgs-dev-env.csh	press Return press Return
4	(If necessary) cleartool setview *ViewName* cd *pathname* cleartool checkout -nc *makefile*	 press Return press Return press Return
5	(No entry)	examine/alter make files
6	smfcompile -*lang* -f *textfile*.t	press Return
7	(No entry)	verify directory structure
8	(If necessary) cleartool checkout -nc *filename*	press Return
9	(No entry)	build the software according to documentation
10	(If necessary) cleartool checkin *filename* -nc	press Return

9.5 Building Science Software with the DAAC Version of the SDP Toolkit

In order to be tested at the DAAC, science software must be compiled and linked to produce binary executables. These binary executables are then packaged into one or more shell scripts as defined by the science software developer (Instrument Team). These science software packages are the Product Generation Executives (PGEs) delivered to the DAACs during SSI&T. PGEs are the smallest schedulable unit of science software in the ECS.

Building science software into PGEs should be done in accordance with supplied documentation. Such documentation should describe the process in detail. In general, science software deliveries will come with make files or other build scripts to automate the build process.

In general, science software will be built, run, and tested with the SCF version of the SDP Toolkit to ensure that the software has been successfully ported to the DAAC. Once this test has been completed successfully, the science software will be re-built, rerun, and re-tested with the DAAC version of the SDP Toolkit. Only with the DAAC Toolkit can the PGE be run within the ECS.

This procedure describes some general principals that may or may not be applicable to a particular science software delivery for building a PGE with the DAAC version of the SDP Toolkit. See Section 9.4 for building a PGE with the SCF version of the SDP Toolkit.

The Activity Checklist table that follows provides an overview of the process for building science software with the DAAC version of the SDP Toolkit. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 9.5-1 Building Science Software with the DAAC Version of the SDP Toolkit - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Read all documentation supplied in the delivery.	(I) 9.5	
2	SSI&T	Log into the SPR SGI.	(I) 9.5	
3	SSI&T	Set up the DAAC version of the SDP Toolkit environment.	(I) 9.5	
4	SSI&T	Set the ClearCase view.	(I) 9.5	
5	SSI&T	Examine and alter (if necessary) any supplied make file.	(I) 9.5	
6	SSI&T	Compile any Status Message Facility (SMF) files	(I) 9.5	
7	SSI&T	Verify the directory structure.	(I) 9.5	
8	SSI&T	If necessary, check out of ClearCase executable or object files.	(I) 9.5	

Table 9.5-1 Building Science Software with the DAAC Version of the SDP Toolkit - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
9	SSI&T	Perform the build.	(I) 9.5	
10	SSI&T	If necessary, check make file(s) back into ClearCase.	(I) 9.5	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative) is the current command shell.

To build science software with the DAAC version of the SDP Toolkit, be aware of the "typical" procedure steps that follow:

1 Read all instructional material supplied with the science software delivery. Such material should be the primary source of information on how to build the science software.

- Read the *Systems Description* document and the *Operations Manual*. Both of these or their equivalent should be in the delivery.
- Typically, there will be "readme" files accompanying each PGE in the directory structure, perhaps in a doc directory.
- Text files (ASCII) may be viewed with the UNIX command, *more* or with the *vi* editor.
- PostScript documents may be viewed with *ghostview*, which is accessible via the SSIT Manager.
- PDF formatted documents may be viewed with *acroread*, the Acrobat Reader, also accessible via the SSIT Manager.
- Documents in Microsoft Word and related formats may be viewed through the Microsoft Windows™ 3.1 emulator. The MS Windows emulator may be accessed from the SSIT Manager.

2 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.

- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.
- It is recommended that this procedure begin within a new command shell on the SPR SGI.

3 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname***, press **Return**. Then type, source **$PGSHOME/bin/*sgiX*/pgs-dev-env.csh**, press **Return**.

- The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version, in this case, a DAAC version (see Section 9.2).
- The ***sgiX*** refers to the appropriate processor (see Section 9.2). For example, type **source $PGSHOME/bin/sgi/pgs-dev-env.csh**, press **Return**.

4 If make files are in ClearCase, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName***, press **Return**. Then, **cd *pathname***, press **Return**. And **cleartool checkout -nc *makefile***, press **Return**.

- The ***ViewName*** is the name of a view allowing the make files to be accessible.
- The ***pathname*** is the full path name of the directory (in the VOB) where the make file has been checked in.
- The ***makefile*** is the name of the make file to examine and possibly modify.
- This step is only necessary if any of the make files (or build scripts) are in ClearCase (in the VOB under configuration management).

5 Examine and alter (if necessary) any make files using any text editor (*vi, emacs*). If the software had already been built and tested with the SCF version of the SDP Toolkit, this step may be unnecessary.

- There may be several make files for a particular PGE.
- Verify that compiler, compiler flag settings, and other environment variable settings are appropriate.
- The Toolkit set up (from step 3) will set many environment variables which can be used in the make files. To see the current environment variable settings, at the UNIX prompt on the SPR SGI, type **env**, press **Return**.

6 Compile any required status message facility (SMF) files and place the header file(s) in the proper directory for building. See Section 9.3.

7 Verify that the directory structure for the PGE source files matches the directory structure expected by the make files or build scripts.

- Deliveries may come with install scripts that place files into various directories according to some predefined structure.

8 If necessary, at the UNIX prompt on the SPR SGI, type **cleartool checkout -nc *filename***, press **Return**.

- The ***filename*** is the file name of the executable, object file, or make file to be checked out of ClearCase. The **-nc** flag means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check out step.
- Note that checking in executable or object files is *not* recommended in the first place.

9 Build the software in accordance with instructions delivered.

- Science software deliveries may come with a single, top-level script to do the entire build or the build process could involve a series of steps, each of which should be described fully in the delivered documentation.

10 If necessary, at the UNIX prompt on the SPR SGI, type **cleartool checkin *filename* -nc**, press **Return**.

- The ***filename*** is the file name of the executable, object file, or make file to be checked into ClearCase. The **-nc** flag means "no comment"; it suppresses ClearCase from prompting for a comment to be associated with the check in step.
- Note that checking in executable or object files is *not* recommended.

Table 9.5-2 Building Science Software with the DAAC Version of the SDP Toolkit - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	(No entry)	read all documentation
2	Tools → Xterm	telnet to SPR SGI
3	setenv PGSHOME *ToolkitPathname* source $PGSHOME/bin/sgi*X*/pgs-dev-env.csh	press Return press Return
4	(If necessary) cleartool setview *ViewName* cd *pathname* cleartool checkout -nc *makefile*	 press Return press Return press Return
5	(No entry)	examine/alter make files
6	smfcompile -*lang* -f *textfile*.t	press Return
7	(No entry)	verify directory structure
8	(If necessary) cleartool checkout -nc *filename*	press Return
9	(No entry)	build the software according to documentation
10	(If necessary) cleartool checkin *filename* -nc	press Return

10. Running a PGE in a Simulated SCF Environment

Science software delivered to the DAACs for SSI&T was developed and tested at individual SCFs using the SCF version of the SDP Toolkit. Before linking the software with the DAAC version of the Toolkit and integrating it with the ECS, it is prudent to first link the software to the SCF version of the Toolkit and run it as was run at the SCF. This type of testing can reveal problems associated with the process of porting the software to another platform whose architecture may be quite different from the one on which the software was developed.

A simulated SCF environment means that the software is built using the SCF version of the Toolkit and is run from the UNIX command line. The Planning and Data Processing System (PDPS) and the IMF Data Server are not involved.

The procedures which follow describe how to run the science software in a simulated SCF environment.

10.1 Setting Up the Environment for Running the PGE

Running a PGE that has been built with the SCF version of the SDP Toolkit requires some environment set up as it does at the SCF. This procedure describes how to set up a simulated SCF environment. Once the environment has been set up, the PGE can be run as described in Section 10.2.

The Activity Checklist table that follows provides an overview of the process for setting up the environment for running the PGE in a simulated SCF environment. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 10.1-1 Setting Up the Environment for Running the PGE - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into the SPR SGI.	(I) 10.1	
2	SSI&T	Set up the SCF version of the SDP Toolkit environment.	(I) 10.1	
3	SSI&T	Set the PGS_PC_INFO_FILE environment variable.	(I) 10.1	

Table 10.1-1 Setting Up the Environment for Running the PGE - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
4	SSI&T	Optionally, remove old log files.	(I) 10.1	
5	SSI&T	If necessary, set any additional environment variables.	(I) 10.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Process Control File (PCF) exists and has been tailored for the DAAC environment (Section 9.1).
2. The C shell or a derivative (*e.g.* T shell) is the current user shell.

To set up an environment for running the PGE, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the SPR SGI.
- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the SPR SGI.
- It is recommended that this procedure begin within a new command shell on the SPR SGI.

2 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname***, press **Return**. Then type, source **$PGSHOME/bin/*sgiX*/pgs-dev-env.csh**, press **Return**.
- The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version, in this case, an SCF version (see Section 9.2).
- The ***sgiX*** refers to the appropriate processor (see Section 9.2). For example, type **source $PGSHOME/bin/sgi/pgs-dev-env.csh**, press **Return**.

3 At the UNIX prompt on the SPR SGI, type **setenv PGS_PC_INFO_FILE *PCFpathname*/*PCFfilename***, press **Return**.
- The ***PCFpathname*** is the full path name to the location of the Process Control File (PCF) to be associated with this PGE.
- The ***PCFfilename*** is the file name of the PCF.
- For example, **setenv PGS_PC_INFO_FILE /disk2/PGE32/PCF/PGE32.pcf**, press **Return**.

4 Optionally, at the UNIX prompt on the SPR SGI, type **rm *LogPathname*/*LogFilename***, press **Return**.
- The ***LogPathname*** is the full path name to the location of the PGE log files for this PGE.
- The ***LogFilename*** is the file name of the PGE log file to remove from a previous run of the same PGE. PGE log files can be Status, User, or Report.

- The ***LogFilename*** may use wildcard characters to remove all of the log files at the same time.
- This step is optional. If log files from a previous run of the same PGE are not removed, they will be appended with the information from the current run.
- The environment will then be set up. Continue on to Section 10.2.

5 If necessary, set any other shell environment variables needed by the PGE by sourcing the appropriate scripts or setting them on the command line.
- For example, for a PGE requiring IMSL, at the UNIX prompt on the SPR SGI, type **source /usr/ecs/Rel_A/COTS/imsl/vni/ipt/bin/iptsetup.cs**, press **Return**.
- For some PGEs, the environment variables to be set will be specified in the documentation or the files to source will be supplied in the delivery. Refer to documentation included in the delivery.

Table 10.1-2 Setting Up the Environment for Running the PGE - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to SPR SGI
2	setenv PGS_PC_INFO_FILE *PCFpathname*/*PCFfilename*	press Return
3	setenv PGSHOME *ToolkitPathname* source $PGSHOME/bin/sgi*X*/pgs-dev-env.csh	press Return press Return
4	(Optional) rm *LogPathname*/*LogFilename*	press Return
5	(If necessary) (No entry)	set any additional environment variables needed

10.2 Running and Profiling the PGE

Profiling a PGE refers to the process of gathering information about the runtime behavior of a PGE. The information includes the wall clock time, user time and system time devoted to the PGE; the amount of memory used; the number of page faults; and the number of input and output blocks.

The Planning and Data Processing System (PDPS) database must be populated with the above information when the PGE is registered with the PDPS during the integration phase of SSI&T. This information may be delivered with the PGE or it may need to be determined at the DAAC during SSI&T. This procedure addresses the latter need.

Note that profiling, as used here, does not involve altering the binary executable to produce instrumented code.

The Activity Checklist table that follows provides an overview of the process for running and profiling the PGE. Column one (**Order**) shows the order in which tasks should be accomplished.

Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 10.2-1 Running and Profiling the PGE - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the location of the PGE binary.	(I) 10.2	
2	SSI&T	Run the PGE with the profiler.	(I) 10.2	
3	SSI&T	Check the exit status of the PGE.	(I) 10.2	
4	SSI&T	Examine the profiling results.	(I) 10.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PGE has been built successfully with the SCF version of the SDP Toolkit (Section 9.4).
2. The required SMF runtime message files have been produced and placed in the correct locations (Section 9.2).
3. The Process Control File (PCF) exists and has been tailored for the DAAC environment (Section 9.1).
4. The required environment for running the PGE has been set up (Section 10.1).
5. The required input files are available and accessible.
6. The C shell or a derivative (*e.g.* T shell) is the current user shell.

To run and profile the PGE, execute the procedure steps that follow:

1 At the UNIX prompt on the SPR SGI in the window containing the set up environment from Section 10.1, type **cd *PGEbinPathname***, press **Return**.
- The ***PGEbinPathname*** is the full path name of the directory containing the built PGE binary executable. For example, **cd /disk2/PGE32/bin/**, press **Return**.

2 At the UNIX prompt on the SPR SGI, type ***DpPrR*****usage *PGE* >& *ResultsOut***, press **Return**.
- The ***PGE*** is the name given to the PGE binary executable.
- The ***ResultsOut*** is the file name in which to capture the profiling results as well as any messages from standard output (stdout) and standard error (stderr) that may

be produced by the running PGE. Note that PGEs should *not* write to stdout or stderr.

- The **DpPrRusage** is the profiling program that outputs information about the runtime behavior of the PGE.
- Depending upon the PGE, it may take some time before the UNIX prompt returns.

3 At the UNIX prompt on the SPR SGI, type **echo $status**, press **Return**.

- The **$status** is an environment variable that stores the exit status of the previous program run, in this case, the PGE.
- A status of zero indicates success; a status of non-zero indicates an error of some kind.
- The meaning of a non-zero exit status should be documented and included with the DAPs.
- This command must be run ***immediately*** after the **DpPrRusage** command.

4 At the UNIX prompt on the SPR SGI, type **vi *ResultsOut***, press **Return**.

- The ***ResultsOut*** is the file name under which the profiling output was saved. Other output of the PGE may also be in this file.
- The **DpPrRusage** results may then be recorded and used when the PGE is registered in the PDPS.
- Any text editor/viewer may be used.

Table 10.2-2 Running and Profiling the PGE - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *PGEbinPathname*	press Return
2	DpPrRusage *PGE* >& *ResultsOut*	press Return
3	echo $status	press Return
4	vi *ResultsOut*	press Return

This page intentionally left blank.

11. PGE Registration and Test Data Preparation

The integration of science software with the ECS requires that information about the Product Generation Executives (PGEs) be made known to the PDPS in its database. In addition, the PGEs themselves and the test files that they use (both input and output) need to be placed on the IMF Data Server. These steps must be accomplished before the science software can be run and tested within the ECS.

11.1 PGE Registration

The procedures in this section describe how to register a new PGE with the ECS. This involves updating the PDPS database with information needed to plan, schedule, and run the PGE. Section 11.1.1 describes how to begin the PGE registration process by creating the necessary ODL files needed for each input and output ESDT used by the PGE. Section 11.1.2 describes how to create an ODL file from the delivered PCF and update the PDPS database with this PGE information. Finally, Section 11.1.3 describes how to supply operational information (resource requirements and runtime statistics) to the PDPS database. This is the last step in the PGE registration process. The order in which these procedures are done is important and should be done as indicated.

The Activity Checklist table that follows provides an overview of the process for running and profiling the PGE. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.1-1 PGE Registration - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Begin PGE registration by determining which ESDTs are needed for the PGE. Verify that an ESDT metadata ODL file exists for each or generate one.	(I) 11.1.1	
2	SSI&T	Using the delivered PCF, create a PGE metadata ODL file for the PGE. Update the PDPS database.	(I) 11.1.2	
3	SSI&T	Supply operational metadata for the PGE via GUI. This completes PGE registration.	(I) 11.1.3	

11.1.1 Updating the PDPS Database with ESDT Metadata

In order to update the PDPS Database with ESDT metadata, the metadata must first be prepared in ODL files, one for each ESDT associated with a particular PGE. Then the SSIT Science Update program must be run (Section 11.1.2). The Activity Checklist below is to be followed for each input and output file for a given PGE.

The Activity Checklist table that follows provides an overview of the process for updating the PDPS database with ESDT metadata. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.1.1-1Updating the PDPS Database with ESDT Metadata - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Determine IMF Data Server ShortName for the ESDT corresponding to the given file.	(I) 11.1.1	
2	SSI&T	Look in ESDT directory for ESDT ODL file. If above file exists, exit this procedure.	(I) 11.1.1	
3	SSI&T	Change directories to location of a working directory.	(I) 11.1.1	
4	SSI&T	Copy the template ESDT metadata ODL file to another file for editing.	(I) 11.1.1	
5	SSI&T	Invoke an editor with the new ESDT metadata ODL file.	(I) 11.1.1	
6	SSI&T	Add the required metadata to the file.	(I) 11.1.1	
7	SSI&T	Save the file and quit the editor.	(I) 11.1.1	
8	SSI&T	Copy the edited ESDT metadata ODL file to the proper directory.	(I) 11.1.1	
9	SSI&T	Repeat steps 1-8 for each ESDT required by PGE. Then continue on to Section 11.1.2.	(I) 11.1.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The environment variable DPAT_ESDT_SCIENCE_MD points to a directory used for containing the PDPS ESDT metadata ODL files. Nominally, this is /usr/ecs/Rel_A/test/data.

To update the PDPS Database with ESDT Metadata, execute the procedure steps that follow:

1 Determine the IMF Data Server ShortName for the ESDT corresponding to the given file. One place to look for ESDT names is the Earth Sciences Online Directory.

2 At a UNIX prompt on an AIT Sun, type **ls $DPAT_ESDT_SCIENCE_MD/ESDT_*ShortName*.odl**, press **Return**.

- The **$DPAT_ESDT_SCIENCE_MD** is an environment variable containing the full path name to the location of existing ESDT ODL files.
- The **ESDT_*ShortName*.odl** is the file name of the ESDT ODL file you are looking for where ***ShortName*** is the ESDT's ShortName. If a file for the desired ESDT is listed, then it has already been prepared and this procedure can be exited now.
- For example, if the desired ESDT has the ShortName ca1182, type **ls $DPAT_ESDT_SCIENCE_MD/ESDT_ca1182.odl,** press **Return**.
- If the desired file is *not* listed, continue on to step 3.

3 At a UNIX prompt on the AIT Sun, type **cd *WorkingPathname***, press **Return.**

- The ***WorkingPathname*** is the full path name to a working directory for which the user has write permissions.
- For example, **cd /home/jdoe/working/**, press **Return**.

4 At a UNIX prompt on the AIT Sun, type **cp /usr/ecs/Rel_A/CUSTOM/data/DPS/ESDT_SHRTNAME.odl.tpl ESDT_*ShortName*.odl**, press **Return**.

- The **ESDT_*ShortName*.odl** is the file name of the ESDT ODL file to be created.
- This command copies a template ESDT ODL file to the ESDT ODL file to be created. The template is well commented.
- For example, type **cp /usr/ecs/Rel_A/CUSTOM/data/DPS/ESDT_SHRTNAME.odl.tpl ESDT_ca1182.odl,** press **Return**.
- The **ESDT_*ShortName*.odl** file naming convention *must* be observed.

5 At a UNIX prompt on the AIT Sun, type **vi ESDT_*ShortName*.odl**, press **Return**.

- The **ESDT_*ShortName*.odl** represents the file name of the ESDT ODL template file created in step 4.
- Any text editor may be used such as *emacs*. For example, **emacs ESDT_ca1182.odl**, press **Return**.

6 In the file, add required metadata to the ODL template.

- Use the internal documentation contained in the ODL file (from the original template) to aid in populating with metadata.
- Note that the ShortName specified within the file must match the ShortName of the file name itself.
- In addition, the ShortNames used in the PDPS PGE metadata ODL file (see Section 11.1.2) must match the ShortNames in these files.

7 Save the changes made to the ESDT metadata ODL file and exit the editor.

- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor's documentation.

8 At a UNIX prompt on the AIT Sun, type **cp ESDT_*ShortName*.odl $DPAT_ESDT_SCIENCE_MD**, press **Return**.

- For example, in the case where the ESDTs ShortName is ca1182, type **cp ESDT_ca1182.odl $DPAT_ESDT_SCIENCE_MD**, press **Return**.
- This copies the newly constructed ESDT metadata ODL file to the proper location.

9 Repeat steps 1 through 8 for each ESDT required by a particular PGE. When all ESDT metadata ODL files have been completed, continue on to Section 11.1.2.

Table 11.1.1-2 Updating the PDPS Database with ESDT Metadata - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	determine IMF Data Server ShortName	(No action)
2	ls $DPAT_ESDT_SCIENCE_MD/ESDT_*ShortName*.odl	press Return
3	cd *WorkingPathname*	press Return
4	cp /usr/ecs/Rel_A/CUSTOM/data/DPS/ ESDT_SHRTNAME.odl.tpl ESDT_*ShortName*.odl	press Return
5	vi ESDT_*ShortName*.odl	press Return
6	add required metadata to file	(No action)
7	:wq	press Return
8	cp ESDT_*ShortName*.odl $DPAT_ESDT_SCIENCE_MD	press Return
9	(No entry)	repeat steps 1-8 for all ESDTs

11.1.2 Updating the PDPS Database with PGE Metadata

In order to update the PDPS Database with PGE metadata, the ESDT metadata ODL files must first be prepared for each ESDT required by the PGE (Section 11.1.1). This section describes how to perform the next step, running the SSIT Science Update program.

The Activity Checklist table that follows provides an overview of the process for updating the PDPS database with PGE metadata. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.1.2-1Updating the PDPS Database with PGE Metadata - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke PCF ODL Template tool.	(I) 11.1.2	
2	SSI&T	Specify the configuration file.	(I) 11.1.2	
3	SSI&T	Specify the Process Control file name.	(I) 11.1.2	
4	SSI&T	Specify the PGE name.	(I) 11.1.2	
5	SSI&T	Specify the PGE version.	(I) 11.1.2	
6	SSI&T	Process another PCF or quit.	(I) 11.1.2	
7	SSI&T	Change directories to location of PCF ODL template file.	(I) 11.1.2	
8	SSI&T	Copy the template file to another file for editing.	(I) 11.1.2	
9	SSI&T	Invoke editor with the copied file.	(I) 11.1.2	
10	SSI&T	Add the required metadata to the file.	(I) 11.1.2	
11	SSI&T	Save the file and quit the editor.	(I) 11.1.2	
12	SSI&T	Copy the edited PGE metadata ODL file to the proper directory.	(I) 11.1.2	
13	SSI&T	Invoke the Science Metadata Update tool.	(I) 11.1.2	
14	SSI&T	Specify the configuration file.	(I) 11.1.2	
15	SSI&T	Specify the mode.	(I) 11.1.2	
16	SSI&T	Specify the PGE name.	(I) 11.1.2	
17	SSI&T	Specify the PGE version.	(I) 11.1.2	
18	SSI&T	Process another ODL file or quit.	(I) 11.1.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager is running.
2. The environment variable DPAT_PGE_SCIENCE_MD points to a directory used for containing the PDPS PGE metadata ODL files. Nominally, this is /usr/ecs/Rel_A/test/output.
3. The required PDPS ESDT metadata ODL files have been prepared and placed in the directory pointed to by DPAT_ESDT_SCIENCE_MD (see Section 11.1.1).

To update the PDPS Database with PGE Metadata, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **PDPS Database** and then **PCF ODL Template**.
- An xterm in which DpAtCreateOdlTemplate.sh is running will be displayed.
- Alternatively, the same tool can be invoked by typing at a UNIX prompt on an AIT Sun **DpAtCreateOdlTemplate.sh**, press **Return**.

2 At the program prompt **Config filename (default *defaultConfigFile*)?**, press **Return**.

- The default configuration file for this tool will be used.
- The ***defaultConfigFile*** will be replaced by the full path name and file name of the default configuration file. The file name will be DpAtCS_*daac*.CFG where *daac* will be replaced by one of {GSFC, EDC, LARC, NSIDC}.

3 At the program prompt **Process Control file name?**, type ***PCFpathname*/*PCFfilename***, press **Return**.

- The ***PCFpathname*** is the full path name to the location of the PCF. If not specified, the directory from which the SSIT Manager was run will be assumed.
- The ***PCFfilename*** is the file name of the PCF.

4 At the program prompt **PGE name?**, type ***PGEname***, press **Return**.

- The ***PGEname*** is the name of the PGE that will be registered.

5 At the program prompt **PGE version (default 1)?**, type ***PGEversion***, press **Return** or just press **Return** if the default shown is correct.

- The ***PGEversion*** is the version of the PGE that will be registered.
- After a brief time, the message "Successfully created ODL template file" should be displayed if the task was successful.
- The program will output a file with the filename **PGE_*PGEname*#*PGEversion*.tpl**.
- For example, if the PGE name was **MOD35**, and the version was **1**, this output file will be named **PGE_MOD35#1.tpl**.

6 At the program prompt **Hit return to run again, 'q <return>' to quit:**, press **Return** to repeat process with another PCF or type **q** and press **Return** to quit.

- The xterm will disappear.

7 At a UNIX prompt on an AIT Sun, type **cd *SSITrunPathname***, press **Return.**

- The ***SSITrunPathname*** is the full path to the directory from which the SSIT Manager was run. This will be the directory where the file **PGE_*PGEname*#*PGEversion*.tpl** will reside.

8 At a UNIX prompt on the AIT Sun, type **cp PGE_*PGEname*#*PGEversion*.tpl PGE_*PGEname*#*PGEversion*.odl**, press **Return.**

- The **PGE_*PGEname*#*PGEversion*.tpl** is the file name of the ODL template file created in step 5.
- The **PGE_*PGEname*#*PGEversion*.odl** is the file name of a copy which can be safely edited. This file name convention must be used.

9 At a UNIX prompt on the AIT Sun, type **vi PGE_*PGEname*#*PGEversion*.odl**, press **Return.**

- The **PGE_*PGEname*#*PGEversion*.odl** is the file name of the copy created in step 8.
- Any text editor may be used such as *emacs*. For example, **emacs MOD35#1.odl**, press **Return**.

10 In the file, add required metadata to the ODL template.

- For an explanation of what metadata is required, see file /usr/ecs/Rel_A/CUSTOM/data/DPS/PGE_ssit#11.tpl.
- Note that the ShortNames typed into this file must each have a corresponding PDPS ESDT metadata ODL file (sec. 11.1.1).
- Note also that two PCF entry objects must be deleted corresponding to logical IDs 10999 and 10250. Delete the entire objects from the file. Further, adjust the CLASS=*n* number so that the remaining PCF entry objects have class numbers in numerical order.
- For PCF entries corresponding to MCFs, add to the object "SCIENCE_GROUP=M*n*" where *n* is an integer (1 for the first MCF, 2 for the second, etc.).
- All objects corresponding to output ESDTs must have the SCIENCE_GROUP and YIELD set.
- All objects corresponding to static input ESDTs must have the SCIENCE_GROUP set, but *not* the YIELD (leave out). Objects corresponding to *dynamic* input ESDTs should not have the SCIENCE_GROUP set.
- See Appendix E for example PCF and corresponding PGE ODL file.

11 Save the changes made to the ODL template file and exit the editor.

- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor's documentation.

12 At a UNIX prompt on the AIT Sun, type **cp PGE_*PGEName#version*.odl $DPAT_PGE_SCIENCE_MD**, press **Return**.

- For example, type **cp MOD35#1.odl $DPAT_PGE_SCIENCE_MD**, press **Return**.
- This copies the newly constructed PGE metadata ODL file to the proper location.

13 From the SSIT Manager, click on the **Tools** menu, then choose **PDPS Database** and then **SSIT Science Metadata Update**.

- An xterm in which DpAtPdpsDbUpdateScience.sh is running will be displayed.
- Alternatively, the same tool can be invoked by typing at a UNIX prompt on an AIT Sun **DpAtPdpsDbUpdateScience.sh**, press **Return**.

14 At the program prompt **Config filename (default *defaultConfigFile*)?**, press **Return**.

- The default configuration file for this tool will be used.
- The ***defaultConfigFile*** will be replaced by the full path name and file name of the default configuration file. The file name will be DpAtDS_*daac*.CFG where *daac* will be replaced by one of {GSFC, EDC, LARC, NSIDC}.

15 At the program prompt **mode (default *defaultMode*)?**, type ***mode***, press **Return** or just press **Return** if the default shown is correct.

- The ***mode*** refers to the database used and will typically be **ops**.

16 At the program prompt **PGE name?**, type ***PGEname***, press **Return**.

- The ***PGEname*** is the name of the PGE that will be registered. This name must match the PGE name specified in step 4.

17 At the program prompt **PGE version (default 1)?**, type ***PGEversion***, press **Return** or just press **Return** if the default shown is correct.

- The ***PGEversion*** is the version of the PGE that will be registered. This version must match the PGE version specified in step 5.
- The PDPS database will then be updated with the information contained in the file **PGE_*PGEname*#*PGEversion*.odl**

18 To the final prompt **Hit return to run again, 'q <return> to quit:**, press **Return** to update the PDPS database with another PGE ODL metadata file or type **q** and press **Return** to quit.

- If you make a mistake entering any values, press **Return** here; your previous entries are restored as defaults and you won't have to retype them.

Table 11.1.2-2 Updating the PDPS Database with PGE Metadata - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → PDPS Database → PCF ODL Template	(No action)
2	(No entry)	press Return
3	*PCFpathname*/*PCFfilename*	press Return
4	*PGEname*	press Return
5	*PGEversion*	press Return
6	q | Return	press Return
7	cd *SSITrunPathname*	press Return
8	cp PGE_*PGEname*#*PGEversion*.tpl PGE_*PGEname*#*PGEversion*.odl	press Return
9	vi PGE_*PGEname*#*PGEversion*.odl	press Return
10	add required metadata to file	(No action)
11	:wq	press Return
12	cp PGE_*PGEName*#*version*.odl $DPAT_PGE_SCIENCE_MD	press Return
13	Tools → PDPS Database → SSIT Science Metadata Update	(No action)
14	(No entry)	press Return
15	*mode*	press Return
16	*PGEname*	press Return
17	*PGEversion*	press Return
18	q | Return	press Return

11.1.3 Updating the PDPS Database with Operational Metadata

The SSIT version of the PDPS database is initialized and updated with SSIT Operational Metadata so that the Planning and Processing Subsystem can schedule and run PGEs. Here, PDPS Operational Metadata refers to PGE information which is supplied to the DAAC/SSIT Operator and may change frequently.

The operator enters this data directly into the SSIT Operational Metadata Update GUI. The program then writes the data directly to the SSIT version of the PDPS database.

Before running the SSIT Operational Metadata Update from either the SSIT Manager or from the command line, you must first update the PDPS with PGE metadata (see Section 11.1.2) and with ESDT metadata (see Section 11.1.1). In addition, to get initial PGE Performance data which will be entered into the GUI, you need to run the profiling utility, DpPrRusage on the PGE or have the information on profiling provided (see Section 10.2).

The Activity Checklist table that follows provides an overview of the process for updating the PDPS database with Operational metadata. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.1.3-1 Updating the PDPS Database with PGE Metadata - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the PDPS/SSIT Database Update GUI.	(I) 11.1.3	
2	SSI&T	Select PGE name and version.	(I) 11.1.3	
3	SSI&T	Choose new PGE.	(I) 11.1.3	
4	SSI&T	Open the PROFILE page of the GUI.	(I) 11.1.3	
5	SSI&T	Enter in performance statistics.	(I) 11.1.3	
6	SSI&T	Enter in resource requirements.	(I) 11.1.3	
7	SSI&T	Update the PDPS database with the information entered.	(I) 11.1.3	
8	SSI&T	Quit the PDPS/SSIT Database Update GUI.	(I) 11.1.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The required UNIX environment variables have been set.
2. The PGE and ESDT metadata have been updated to the PDPS database for this PGE (Sections 11.1.1 and 11.1.2).

3. The SSIT Manager's internal PCF entry that calls this tool has the correct database name in the argument list. That is, the relevant entry looks like:

   ```
   630|DpCAtMgrSSITOpnlMdUp|DpAtOpDbGui ConfigFile
   /usr/ecs/Rel_A/CUSTOM/cfg/DpAtDO_daac.CFG ecs_mode mode
   ```

   where *daac* is one of {GSFC, EDC, LARC, NSIDC} and the *mode* refers to the correct database (typically set to ssit). For example,

   ```
   630|DpCAtMgrSSITOpnlMdUp|DpAtOpDbGui ConfigFile
   /usr/ecs/Rel_A/CUSTOM/cfg/DpAtDO_GSFC.CFG ecs_mode ssit
   ```

To update the SSIT version of the PDPS database with operational metadata, execute the steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **PDPS Database** and then **SSIT Opnl Metadata Update**.
- The PDPS/SSIT Database Update GUI will be displayed.

2 Click on the radio button labeled **NEW PGE** in the lower left quadrant.
- The page tabs **PROFILE**, **RUNTIME**, and **ESDT** will change from grey (indicating disabled) to black (indicating enabled).

3 In the subwindow labeled **PGE Names**, click on a PGE name. Then in the subwindow labeled **PGE Versions**, click on the PGE version for that PGE. Then click on the button labeled **DONE**.
- The PGE name and version will be highlighted when you click on them.
- If the PGE name and/or version does not appear in the lists, it means that the updating of the PDPS database with PGE metadata was not successful.

4 Click on the **PROFILE** page tab.
- The Profile page will be displayed.

5 In the fields under the label **Performance Statistics**, enter the information specified.
- In the field labeled **Wall clock time**, enter the amount of wall clock time it takes for one execution of the PGE, in seconds. The tab **PROFILE** will change from black (indicating enabled) to red (indicating database needs to be updated by APPLY button).
- In the field labeled **CPU time (user)**, enter the so-called *user* time of the PGE, in seconds. This value should come from profiling the PGE (see Section 10.2).
- In the field labeled **Max memory used**, enter the maximum amount of memory used by the PGE, in megabytes (MB). This value should come from profiling the PGE (see Section 10.2).
- In the fields labeled **Block input ops** and **Block output ops**, enter the integer number of block inputs and block outputs, respectively. These values should come from profiling the PGE (see Section 10.2).

- In the field labeled **Swaps**, enter the integer number of page swaps from the PGE. This value should come from profiling the PGE (see Section 10.2).
- In the field labeled **Page faults**, enter the integer number of page faults from the PGE. This value should come from profiling the PGE (see Section 10.2).

6 In the fields under the label **Resource Requirements**, enter the information specified.

- In the field labeled **Max. DISK SPACE used during PGE run**, enter the maximum amount of disk used by the PGE during execution, in megabytes (MB). Space should be allowed for the executable(s), input files, output files, ancillary files, static files, MCFs, and the PCF. (This number should also be in the PGE metadata ODL file; yes, there is duplication here.)
- Click on the radio button labeled **Proc. String** (if not already clicked on).
- A list of processing strings should appear in the scrollable window to the left of the two radio buttons **Proc. String** and **Computer Name**. Nominally, only one item should be listed and should be highlighted.
- In the field labeled **No. processors**, the number 1 should appear. In the Testbed, this number is always 1, and cannot be changed. This is a placeholder for Release B.0 .

7 Once the fields on the **PROFILE** page have been completed, click on the **APPLY** button.

- This will update the PDPS database with the information just entered. The tab **PROFILE** will change from red (indicating database needs to be updated) to black (indicating enabled).
- An information box will be displayed; click on **Ok**.
- To start over, click on the **RESET** button. This will clear all fields.

8 Click on the **File** menu and select **Exit**.

- This will end the session with PDPS/SSIT Database Update and the GUI will disappear.

Table 11.1.3-2 Updating the PDPS Database with Operational Metadata - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → PDPS Database → SSIT Opnl Metadata Update	(No action)
2	NEW PGE	click button
3	*PGE name* *PGE version* DONE	click click click button
4	PROFILE	click tab

Table 11.1.3-2 Updating the PDPS Database with Operational Metadata - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
5	*Wall clock time* *CPU time (user)* *Max memory used* *Block input ops* *Block output ops* *Swaps* *Page faults*	(No action)
6	Max. DISK SPACE used during PGE run Proc. String	
7	APPLY	click button
8	File → Exit	(No action)

11.2 Test Data Preparation

This section describes how to prepare test data for use by registered PGEs. When PGEs are first delivered to the DAAC and registered within the PDPS, they will typically be run in isolation. That is, they will be run without any PGE dependencies. For this testing to be possible, test input data granules required by the PGE need to be pre-Inserted to the IMF Data Server.

Data granules can be *dynamic* or *static*. Dynamic data granules are those whose temporal locality differs for each instance of the granule. Examples of dynamic granules are Level 0, Level 1, and Level 2 data sets. Static data granules are those whose temporal locality is static over long periods of time. Examples of static granules are calibration files which may only change with a new version of a PGE. For any granule to be Inserted to the IMF Data Server, a Target MCF is needed (also known as an ASCII metadata ODL file).

In the actual production environment, a Target MCF is produced by the PGE during execution. Thus, the data granule can be Inserted. In isolation testing of a PGE, however, the inputs needed by it will not have been Inserted by a previous PGE in the chain. This Insertion must be done manually. Section 11.2.1 describes how to use the delivered Source MCF for a dynamic data granule to create a Target MCF directly, without a PGE. Section 11.2.2 then describes how to do the Insert. In this way, a dynamic data granule can be Inserted to the IMF as if a PGE had produced it.

Sections 11.3.1 and 11.3.2 describe how to accomplish similar tasks for static data granules. In many cases, static granules will not have an associated Source MCF delivered from which to create a Target MCF. Hence, a Target MCF will need to be generated from scratch. Fortunately, static data granules require very little granule level metadata, thus making this task relatively easy.

The Activity Checklist table that follows provides an overview of the process for test data preparation. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.2-1 Test Data Preparation - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Determine which inputs needed by a PGE are static and which are dynamic.	(I) 11.1.3	
2	SSI&T	For each dynamic data granule, create a Target MCF using the delivered Source MCF. Hand edit the Target MCF to supply metadata that normally would have been supplied by a PGE.	(I) 11.2.1	
3	SSI&T	Insert each dynamic data granule to the IMF Data Server.	(I) 11.2.2	
4	SSI&T	For each static data granule, create a Target MCF using a Target MCF template.	(I) 11.2.3	
5	SSI&T	Insert each static data granule to the IMF Data Server.	(I) 11.2.4	

11.2.1 Creating a Target MCF for a Dynamic Granule

A Target MCF file for a corresponding dynamic data granule can be created based on the information provided in the Source MCF file and the involved science software package (PGE). Since the Target MCF file for a dynamic data granule is similar to the Source MCF file for the same data granule which is normally available from the delivered science software package, a program to convert a Source MCF file to a Target MCF file has been developed. The execution of this conversion program will create a template Target MCF file from a Source MCF file and after a minor editing, this file will be used as input to the IMF Data Server when the dynamic data granule is Inserted. The following procedure describes how to generate this Target MCF file from a given Source MCF file.

The Activity Checklist table that follows provides an overview of the process for creating a metadata ODL file for a dynamic granule. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.2.1-1 Creating a Target MCF for a Dynamic Granule - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to locate the Source MCF file	(I) 11.2.1	
2	SSI&T	Invoke Source to Target MCF template program	(I) 11.2.1	
3	SSI&T	Specify the Source MCF file name	(I) 11.2.1	
4	SSI&T	Specify the Target MCF template file name with the file name extension **.met**	(I) 11.2.1	
5	SSI&T	Bring up the *targetMCFname*.met in a text editor (e.g. *vi*)	(I) 11.2.1	
6	SSI&T	Edit the Target MCF template file	(I) 11.2.1	
7	SSI&T	Save the changes made and Quit *vi* editor	(I) 11.2.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. A Source MCF (the MCF delivered with the PGE) has been properly created for the corresponding granule and is available for reference.

To create a Target MCF for a dynamic granule, execute the steps that follow:

1 At the UNIX prompt on an AIT Sun, type **cd *SourceMCFpathname***, press **Return**.
- The ***SourceMCFpathname*** is the full path name of the directory containing the Source MCF file for the dynamic data granule to be inserted into the IMF Data Server. For example, **cd /Rel_A/data/SourceMCF/**, press **Return**.

2 At the UNIX prompt on the AIT Sun, type **SrcToTargetMCF**, press **Return**
- This will invoke the Source to Target MCF template program and prompt the user for further information (see steps 3 - 7).

3 At the prompt **Enter source MCF file name:** type ***SourceMCFfilename***, press **Return**
- The ***SourceMCFfilename*** is the file name of the Source MCF for the data granule to be Inserted. For example, type **CER01T_1.MCF**, press **Return**.

4 At the prompt **Enter target MCF filename:** type ***TargetMCFfilename*.met**, press **Return**.
- The ***TargetMCFfilename*** is the file name of the Target MCF template corresponding to the data granule to be Inserted. The **.met** file name extension is required for the Target MCF file. For example, type **CER01T_1.met**, press **Return**.

5 At the UNIX prompt on the AIT Sun, using a text editor to edit the Target MCF file template. If *vi* text editor is used, type **vi *TargetMCFfileame.met***, press **Return**.

- This command invokes the *vi* editor and reads in the ***TargetMCFfileame.met*** template file. For example, type **vi CER01T_1.met**, press **Return**.
- For other text editors, refer to that editor's documentation.

6 Edit the Target MCF template in accordance with the information provided from the science software (PGE). The following guidelines should be followed when performing the edition on the Target MCF template file:
  - For those objects with the **Data_Location="PGE"** in the Source MCF, such as for those describe the date, time, and the file locations, the resulting data values for the corresponding Target MCF file must be filled out by the SSIT operator. These data values should be obtained from the instrument team based on the specific PGE requirements.
  - A sample Source MCF file and the resulting Target MCF file are shown in Listings 11.2.1-1 and 11.2.1-2, respectively.

7 Save the changes made to the Target MCF template and exit the editor. Type **:wq**, press **Return**.
  - This will create a Target MCF file for the dynamic data granule to be inputted into the IMS Data Server.

Table 11.2.1-2 Creating a Target MCF for a Dynamic Granule - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *SourceMCFpathname*	(No action)
2	SrcToTargetMCF	(No action)
3	*SourceMCFfilename*,	(No action)
4	*TargetMCFfilename*.met_	(No action)
5	vi *TargetMCFfilename*.met	enter specific values for VALUE= " " in the Target MCF template file if they are blank.
6	:wq	(No action)

If the Source MCF file is not available such as in the case when the data granule is used for the first PGE in the chain process, the Target MCF can be generated by editing an existing Target MCF file for another data granule using the information provided by the involved instrument team.

Listing 11.2.1-1 - Sample Source MCF

```
GROUP = INVENTORYMETADATA


GROUPTYPE = MASTERGROUP


      OBJECT=SHORTNAME

            DATA_LOCATION="MCF"

            TYPE="STRING"

            NUM_VAL=1

            VALUE="MOD29"

            MANDATORY="TRUE"

      END_OBJECT=SHORTNAME


      OBJECT=VERSIONID

            Data_Location = "MCF"

            Value="1"

            TYPE="STRING"

            NUM_VAL=1

            Mandatory ="TRUE"

      END_OBJECT=VERSIONID


      OBJECT=SIZEMBECSDATAGRANULE

            DATA_LOCATION="PGE"

            TYPE="FLOAT"

            NUM_VAL=1

            MANDATORY="TRUE"

      END_OBJECT=SIZEMBECSDATAGRANULE
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
OBJECT=OPERATIONMODE
      DATA_LOCATION="PGE"
      TYPE="STRING"
      NUM_VAL=1
      MANDATORY="TRUE"
END_OBJECT=OPERATIONMODE

GROUP = BOUNDINGRECTANGLE

      OBJECT=EASTBOUNDINGCOORDINATE
            DATA_LOCATION="PGE"
            TYPE="DOUBLE"
            NUM_VAL=1
            MANDATORY="TRUE"
      END_OBJECT=EASTBOUNDINGCOORDINATE

      OBJECT=WESTBOUNDINGCOORDINATE
            DATA_LOCATION="PGE"
            TYPE="DOUBLE"
            NUM_VAL=1
            MANDATORY="TRUE"
      END_OBJECT=WESTBOUNDINGCOORDINATE

      OBJECT=NORTHBOUNDINGCOORDINATE
            DATA_LOCATION="PGE"
            TYPE="DOUBLE"
            NUM_VAL=1
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
            MANDATORY="TRUE"
      END_OBJECT=NORTHBOUNDINGCOORDINATE

      OBJECT=SOUTHBOUNDINGCOORDINATE
            DATA_LOCATION="PGE"
            TYPE="DOUBLE"
            NUM_VAL=1
            MANDATORY="TRUE"
      END_OBJECT=SOUTHBOUNDINGCOORDINATE

END_GROUP = BOUNDINGRECTANGLE

GROUP = GPOLYGON

      OBJECT=GRingContainer
            Data_Location="NONE"
            CLASS="M"
            Mandatory="TRUE"

            OBJECT=EXCLUSIONGRINGFLAG
                  Data_Location="PGE"
                  CLASS="M"
                  TYPE="STRING"
                  NUM_VAL=1
/*                Value="N"         */
                  Mandatory="FALSE"
            END_OBJECT=EXCLUSIONGRINGFLAG
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
      OBJECT=GRINGPOINTLATITUDE
            Data_Location="PGE"
            CLASS="M"
            TYPE="DOUBLE"
            NUM_VAL=4
            Mandatory="FALSE"
      END_OBJECT=GRINGPOINTLATITUDE

      OBJECT=GRINGPOINTLONGITUDE
            Data_Location="PGE"
            CLASS="M"
            TYPE="DOUBLE"
            NUM_VAL=4
            Mandatory="FALSE"
      END_OBJECT=GRINGPOINTLONGITUDE

      OBJECT=GRINGPOINTSEQUENCENO
            Data_Location="PGE"
            CLASS="M"
            TYPE="INTEGER"
            NUM_VAL=4
            Mandatory="FALSE"
      END_OBJECT=GRINGPOINTSEQUENCENO

END_OBJECT=GRingContainer
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
END_GROUP = GPOLYGON


GROUP = ORBITCALCULATEDSPATIALDOMAIN


      OBJECT=ORBITNUMBER
            DATA_LOCATION="PGE"
            TYPE="INTEGER"
            NUM_VAL=1
            MANDATORY="TRUE"
      END_OBJECT=ORBITNUMBER


END_GROUP = ORBITCALCULATEDSPATIALDOMAIN


GROUP = RANGEDATETIME


      OBJECT=RANGEBEGINNINGDATETIME
            DATA_LOCATION="PGE"
            TYPE="DATETIME"
            NUM_VAL=1
            MANDATORY="TRUE"
      END_OBJECT=RANGEBEGINNINGDATETIME


      OBJECT=RANGEENDINGDATETIME
            DATA_LOCATION="PGE"
            TYPE="DATETIME"
            NUM_VAL=1
            MANDATORY="TRUE"
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
      END_OBJECT=RANGEENDINGDATETIME


END_GROUP = RANGEDATETIME


GROUP = QASTATS


      OBJECT=QAPERCENTINTERPOLATEDDATA

            DATA_LOCATION="PGE"

            TYPE="INTEGER"

            NUM_VAL=1

            MANDATORY="TRUE"

      END_OBJECT=QAPERCENTINTERPOLATEDDATA


      OBJECT=QAPERCENTOUTOFBOUNDSDATA

            DATA_LOCATION="PGE"

            TYPE="INTEGER"

            NUM_VAL=1

            MANDATORY="TRUE"

      END_OBJECT=QAPERCENTOUTOFBOUNDSDATA


      OBJECT=QAPERCENTMISSINGDATA

            DATA_LOCATION="PGE"

            TYPE="INTEGER"

            NUM_VAL=1

            MANDATORY="TRUE"

      END_OBJECT=QAPERCENTMISSINGDATA
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
END_GROUP = QASTATS


GROUP = QACOLLECTIONSTATS


      OBJECT=AUTOMATICQUALITYFLAG
            DATA_LOCATION="PGE"
            TYPE="STRING"
            NUM_VAL=1
            MANDATORY="TRUE"
      END_OBJECT=AUTOMATICQUALITYFLAG

      OBJECT=OPERATIONALQUALITYFLAG
            DATA_LOCATION="MCF"
            TYPE="STRING"
            NUM_VAL=1
            MANDATORY="TRUE"
            VALUE="not being investigated in this 5.1 attempt"
      END_OBJECT=OPERATIONALQUALITYFLAG

      OBJECT=SCIENCEQUALITYFLAG
            DATA_LOCATION="MCF"
            TYPE="STRING"
            NUM_VAL=1
            MANDATORY="TRUE"
            VALUE="not being investigated"
      END_OBJECT=SCIENCEQUALITYFLAG
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
      OBJECT=QUALITYFLAGEXPLANATION

            DATA_LOCATION="PGE"

            TYPE="STRING"

            NUM_VAL=1

            MANDATORY="TRUE"

      END_OBJECT=QUALITYFLAGEXPLANATION


END_GROUP = QACOLLECTIONSTATS


GROUP = ECSDATAGRANULE


      OBJECT=REPROCESSINGACTUAL

            DATA_LOCATION="MCF"

            TYPE="STRING"

            NUM_VAL=1

            VALUE="processed once"

            MANDATORY="TRUE"

      END_OBJECT=REPROCESSINGACTUAL


      OBJECT=REPROCESSINGPLANNED

            DATA_LOCATION="MCF"

            TYPE="STRING"

            NUM_VAL=1

            VALUE="no further update anticipated"

            MANDATORY="TRUE"

      END_OBJECT=REPROCESSINGPLANNED
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
    END_GROUP = ECSDATAGRANULE

    GROUP = INPUTGRANULE

          OBJECT=InputPointerContainer
                Data_Location = "NONE"
                CLASS="M"
                Mandatory="TRUE"

                OBJECT=INPUTPOINTER
                      Data_Location="PGE"
                      TYPE="STRING"
                      CLASS="M"
                      NUM_VAL=1
                      Mandatory="FALSE"
                END_OBJECT=INPUTPOINTER

          END_OBJECT=InputPointerContainer

    END_GROUP = INPUTGRANULE

END_GROUP = INVENTORYMETADATA

GROUP = ARCHIVEDMETADATA

GROUPTYPE = MASTERGROUP
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
OBJECT=ALGORITHMPACKAGEACCEPTANCEDATE
      DATA_LOCATION="MCF"
      TYPE="STRING"
      NUM_VAL=1
      MANDATORY="TRUE"
      VALUE="1997-01-01"
END_OBJECT=ALGORITHMPACKAGEACCEPTANCEDATE

OBJECT=ALGORITHMPACKAGEMATURITYCODE
      DATA_LOCATION="MCF"
      TYPE="STRING"
      NUM_VAL=1
      MANDATORY="TRUE"
      VALUE="pre-launch"
END_OBJECT=ALGORITHMPACKAGEMATURITYCODE

OBJECT=ALGORITHMPACKAGENAME
      DATA_LOCATION="MCF"
      VALUE="V1 MOD29(SEAICE) L2"
      TYPE="STRING"
      NUM_VAL=1
      MANDATORY="TRUE"
END_OBJECT=ALGORITHMPACKAGENAME

OBJECT=ALGORITHMPACKAGEVERSION
      DATA_LOCATION="MCF"
      TYPE="STRING"
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
      NUM_VAL=1
      MANDATORY="TRUE"
      VALUE="version 1"
END_OBJECT=ALGORITHMPACKAGEVERSION

OBJECT=INSTRUMENTNAME
      DATA_LOCATION="MCF"
      TYPE="STRING"
      NUM_VAL=1
      MANDATORY="TRUE"
      VALUE="Moderate-Resolution Imaging SpectroRadiometer"
END_OBJECT=INSTRUMENTNAME

OBJECT=PLATFORMSHORTNAME
      DATA_LOCATION="MCF"
      TYPE="STRING"
      NUM_VAL=1
      MANDATORY="TRUE"
      VALUE="EOS AM1"
END_OBJECT=PLATFORMSHORTNAME

OBJECT=PROCESSINGCENTER
      DATA_LOCATION="MCF"
      VALUE="GSFC"
      TYPE="STRING"
      NUM_VAL=1
      MANDATORY="TRUE"
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
END_OBJECT=PROCESSINGCENTER


OBJECT=GRANULENUMBER

      DATA_LOCATION="PGE"

      TYPE="INTEGER"

      NUM_VAL=1

      MANDATORY="TRUE"

END_OBJECT=GRANULENUMBER


OBJECT=LONGNAME

      DATA_LOCATION="MCF"

      VALUE="MOD29 L2 SEAICE COVER GRANULE"

      TYPE="STRING"

      NUM_VAL=1

      MANDATORY="TRUE"

END_OBJECT=LONGNAME


OBJECT=MODISPRODUCTFILENAME

      DATA_LOCATION="PGE"

      TYPE="STRING"

      NUM_VAL=1

      MANDATORY="TRUE"

END_OBJECT=MODISPRODUCTFILENAME


GROUP = SPSO


      OBJECT=SPSOPARAMETERS
```

Listing 11.2.1-1 - Sample Source MCF (cont.)

```
                        DATA_LOCATION="PGE"

                        TYPE="STRING"

                        NUM_VAL=1

                        MANDATORY="TRUE"

                END_OBJECT=SPSOPARAMETERS


        END_GROUP=SPSO


        OBJECT=PROCESSINGDATETIME

                DATA_LOCATION="PGE"

                TYPE="STRING"

                NUM_VAL=1

                MANDATORY="TRUE"

        END_OBJECT=PROCESSINGDATETIME


END_GROUP=ARCHIVEDMETADATA


END
```

Listing 11.2.2-1 - Sample Corresponding Target MCF

```
GROUP= INVENTORYMETADATA

GROUPTYPE= MASTERGROUP

OBJECT= SHORTNAME

NUM_VAL= 1

VALUE= "MOD29"

END_OBJECT= SHORTNAME

OBJECT= VERSIONID

VALUE= "1"

NUM_VAL= 1

END_OBJECT= VERSIONID

OBJECT= SIZEMBECSDATAGRANULE

VALUE =

NUM_VAL= 1

END_OBJECT= SIZEMBECSDATAGRANULE
```

Listing 11.2.2-1 - Sample Corresponding Target MCF (cont.)

```
OBJECT= OPERATIONMODE

VALUE =

NUM_VAL= 1

END_OBJECT= OPERATIONMODE

GROUP= BOUNDINGRECTANGLE

OBJECT= EASTBOUNDINGCOORDINATE

VALUE =

NUM_VAL= 1

END_OBJECT= EASTBOUNDINGCOORDINATE

OBJECT= WESTBOUNDINGCOORDINATE

VALUE =

NUM_VAL= 1

END_OBJECT= WESTBOUNDINGCOORDINATE
```

Listing 11.2.2-1 - Sample Corresponding Target MCF (cont.)

```
OBJECT= NORTHBOUNDINGCOORDINATE

VALUE =

NUM_VAL= 1

END_OBJECT= NORTHBOUNDINGCOORDINATE

OBJECT= SOUTHBOUNDINGCOORDINATE

VALUE =

NUM_VAL= 1

END_OBJECT= SOUTHBOUNDINGCOORDINATE

END_GROUP= BOUNDINGRECTANGLE

GROUP= GPOLYGON

OBJECT= GRingContainer

OBJECT= EXCLUSIONGRINGFLAG

VALUE =

NUM_VAL= 1
```

Listing 11.2.2-1 - Sample Corresponding Target MCF (cont.)

```
END_OBJECT= EXCLUSIONGRINGFLAG

OBJECT= GRINGPOINTLATITUDE

VALUE =

NUM_VAL= 4

END_OBJECT= GRINGPOINTLATITUDE

OBJECT= GRINGPOINTLONGITUDE

VALUE =

NUM_VAL= 4

END_OBJECT= GRINGPOINTLONGITUDE

OBJECT= GRINGPOINTSEQUENCENO

VALUE =

NUM_VAL= 4

END_OBJECT= GRINGPOINTSEQUENCENO

END_OBJECT= GRingContainer
```

Listing 11.2.2-1 - Sample Corresponding Target MCF (cont.)

```
END_GROUP= GPOLYGON

GROUP= ORBITCALCULATEDSPATIALDOMAIN

OBJECT= ORBITNUMBER

VALUE =

NUM_VAL= 1

END_OBJECT= ORBITNUMBER

END_GROUP= ORBITCALCULATEDSPATIALDOMAIN

GROUP= RANGEDATETIME

OBJECT= RANGEBEGINNINGDATETIME

VALUE =

NUM_VAL= 1

END_OBJECT= RANGEBEGINNINGDATETIME

OBJECT= RANGEENDINGDATETIME

VALUE =
```

Listing 11.2.2-1 - Sample Corresponding Target MCF (cont.)

```
NUM_VAL= 1

END_OBJECT= RANGEENDINGDATETIME

END_GROUP= RANGEDATETIME

GROUP= QASTATS

OBJECT= QAPERCENTINTERPOLATEDDATA

VALUE =

NUM_VAL= 1

END_OBJECT= QAPERCENTINTERPOLATEDDATA

OBJECT= QAPERCENTOUTOFBOUNDSDATA

VALUE =

NUM_VAL= 1

END_OBJECT= QAPERCENTOUTOFBOUNDSDATA

OBJECT= QAPERCENTMISSINGDATA

VALUE =
```

Listing 11.2.2-1 - Sample Corresponding Target MCF (cont.)

```
NUM_VAL= 1

END_OBJECT= QAPERCENTMISSINGDATA

END_GROUP= QASTATS

GROUP= QACOLLECTIONSTATS

OBJECT= AUTOMATICQUALITYFLAG

VALUE =

NUM_VAL= 1

END_OBJECT= AUTOMATICQUALITYFLAG

OBJECT= OPERATIONALQUALITYFLAG

NUM_VAL= 1

VALUE= "notbeinginvestigatedinthis5.1attempt"

END_OBJECT= OPERATIONALQUALITYFLAG

OBJECT= SCIENCEQUALITYFLAG

NUM_VAL= 1
```

Listing 11.2.2-1 - Sample Corresponding Target MCF (cont.)

```
VALUE= "notbeinginvestigated"

END_OBJECT= SCIENCEQUALITYFLAG

OBJECT= QUALITYFLAGEXPLANATION

VALUE =

NUM_VAL= 1

END_OBJECT= QUALITYFLAGEXPLANATION

END_GROUP= QACOLLECTIONSTATS

GROUP= ECSDATAGRANULE

OBJECT= REPROCESSINGACTUAL

NUM_VAL= 1

VALUE= "processedonce"

END_OBJECT= REPROCESSINGACTUAL

OBJECT= REPROCESSINGPLANNED

NUM_VAL= 1
```

Listing 11.2.2-1 - Sample Corresponding Target MCF (cont.)

```
VALUE= "nofurtherupdateanticipated"

END_OBJECT= REPROCESSINGPLANNED

END_GROUP= ECSDATAGRANULE

GROUP= INPUTGRANULE

OBJECT= InputPointerContainer

OBJECT= INPUTPOINTER

VALUE =

NUM_VAL= 1

END_OBJECT= INPUTPOINTER

END_OBJECT= InputPointerContainer

END_GROUP= INPUTGRANULE

END
```

11.2.2 Inserting Dynamic Data Granules to the IMF Data Server

In order for dynamic data files to be used both during the SSI&T and in production, this file must exist in the IMF Data Server and be accessible by the local machine. A program called the Insert Test Dynamic File can be used for Inserting a dynamic data granule into the IMF Data Server.

The Activity Checklist table that follows provides an overview of the process for Inserting a dynamic data granule into the IMF Data Server. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.2.2-1 Inserting Dynamic Data Granules to the IMF Data Server - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the Insert Test Dynamic File tool.	(I) 11.2.2	
2	SSI&T	Specify configuration file name.	(I) 11.2.2	
3	SSI&T	Specify ESDT ShortName.	(I) 11.2.2	
4	SSI&T	Specify file name of granule to Insert.	(I) 11.2.2	
5	SSI&T	Specify Target MCF file name.	(I) 11.2.2	
6	SSI&T	Specify quit or continue with other Inserts.	(I) 11.2.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ESDT descriptor file for the granule to be Inserted exists in the ODL format on the IMF Data Server.
2. The Target MCF for this data granule has been created for the Insert.

To Insert a dynamic granule to the IMF Data Server, execute the following steps:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Data Server** and then **Insert Test Dynamic**.

- An xterm in which DpAtInsertTestFile.sh is running will be displayed.
- Alternatively, the same tool can be invoked by typing at a UNIX prompt on an AIT Sun **DpAtInsertTestFile.sh**, press **Return**.
- The DpAtInsertTestFile.sh program will prompt for further information.

2 At the program prompt **Config filename (default *defaultConfigFile*)?**, press **Return**.

- The default configuration file for this tool will be used.

- The ***defaultConfigFile*** will be replaced by the full path name and file name of the default configuration file. The file name will be DpAtID_*daac*.CFG where *daac* will be replaced by one of {GSFC, EDC, LARC, NSIDC}.

3 At the program prompt **ESDT name?** type ***ESDTShortName***, press **Return**
- The ***ESDTShortName*** is the ShortName of the ESDT descriptor file corresponding to this granule to be Inserted. For example, type **CER01T**, press **Return**.

4 At the program prompt **Filename to insert?** type ***pathname*/*GranuleFilename***, press **Return**
- The ***pathname*/*GranuleFileName*** is the full path name and file name of the data granule to be Inserted. For example, type **/data/CERES/CER01T_1**, press **Return**.

5 At the program prompt **Associated ASCII metadata (target MCF) filename to insert (default *pathname*/*GranuleFileName*.met)?**, press **Return**.
- The default is the file name of the granule to insert with the .met file name extension. If the default is not correct, then the file name of this file must be entered.
- The dynamic data granule will be Inserted to the IMF Data Server. For reference, the IMF Data Server Universal Reference (UR) will be printed on the screen.

6 At the program prompt **Hit return to run again, 'q <return>' to quit:** type **q** and press **Return** to quit or just press **Return** to insert additional dynamic granules.
- If continuing, repeat steps 2 through 5.

Table 11.2.2-2 Inserting Dynamic Data Granules to the IMF Data Server - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Data Server → Insert Test Dynamic	(No action)
2	(No entry)	press Return
3	*ESDTShortName*	press Return
4	*pathname*/*GranuleFilename*	press Return
5	(No entry)	press Return
6	q | Return	press Return

11.2.3 Creating a Target MCF for a Static Granule

In order to Insert a static data granule into the IMF Data Server, a corresponding Target MCF must be generated for the static data granule. Such a metadata MCF file can be obtained by editing an existing template ODL file with the information for the specific data granule. The following procedure describes how to generate this metadata ODL file for a static data granule file.

The Activity Checklist table that follows provides an overview of the process for creating a metadata ODL file for a static granule. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.2.3-1 Creating a Target MCF for a Static Granule - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the working directory	(I) 11.2.3	
2	SSI&T	Copy the template Target MCF to a new Target MCF.	(I) 11.2.3	
3	SSI&T	Invoke text editor with the new Target MCF.	(I) 11.2.3	
4	SSI&T	Specify the values in the blank areas in the template ODL file	(I) 11.2.3	
5	SSI&T	Save the changes and quit the editor	(I) 11.2.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. Sufficient information for the static data granule file is available for reference.

To create a metadata ODL file for a static granule, execute the steps that follow:

1 At the UNIX prompt on an AIT Sun, type **cd *WorkingPathname***, press **Return**.
- The ***WorkingPathname*** is the full path name of the working directory containing the template metadata ODL file. For example, **cd /Rel_A/data/TemplateODL/**, press **Return**.

2 At the UNIX prompt on the AIT Sun, type **cp *StaticODLmet.tpl filename*.met**, press **Return**.
- The ***StaticODLmet.tpl*** is the file name of the template Target MCF.
- The ***filename*.met** is the file name of the Target MCF for this static file. The file name extension must be .met.
- This command will copy the template Target MCF to ***filename*.met**. For example, type **cp StaticODLmet.tpl CER11T.mcf.met**, press **Return**.

3 At a UNIX prompt on the AIT Sun, type **vi *filename*.met**, press **Return.**
- This command invokes the *vi* editor and reads in the Target MCF created in step 2.

- Any text editor may be used such as *emacs*. For example, **emacs CER11T.met**, press **Return**.

4 Edit the Target MCF with the specific information for the static data granule to be Inserted. The following guidelines should be followed when editing on the template MCF:
  - The value for the SHORTNAME object should be filled out with proper instrument name. For example: “ CER_MISC”.
  - The value for the VERSIONID object should be filled out with the proper version number. For example: “1” .
  - In the INFORMATIONCONTENTCONTAINER object,
    - The value for the PARAMETERNAME object of the class “1” should be filled out with the name of static data file. For example: “CER11T.mcf”.
    - The value for the PARAMETERVALUE object of the class “2” should be filled out based on the following guideline:
      - If the data granule is a coefficient file, a “C” followed by a numerical number n (n=1,2,….) will be used. Here n stands for the number of the coefficient file. For example, if the first coefficient file is involved, “C1” will be used.
      - If the data granule is a MCF file, a “M” followed by a numerical number n (n=1,2,….) will be used. Here n stands for the number of the MCF file. For example, if the first MCF file is involved, “M1” will be used.
  - A template Target MCF and a sample Target MCF are shown in Listings 11.3.1-1 and 11.3.1-2, respectively.

5 Save the changes made to the Target MCF (***filename*.met**) and exit the editor.
- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor’s documentation.

Table 11.2.3-2 Creating a Target MCF for a Static Granule - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *WorkingPathname*	press Return
2	cp StaticODLmet.plt *filename*.met	press Return
3	vi *filename*.met	press Return
4	edit the file in the editor	(No action)
5	:wq	press Return

Listing 11.2.3-1 - Target MCF Template for a Static Granule

```
GROUP= INVENTORYMETADATA
      GROUPTYPE= MASTERGROUP

      OBJECT= SHORTNAME
            NUM_VAL= 1
            VALUE= "         "
      END_OBJECT= SHORTNAME

      OBJECT= VERSIONID
            NUM_VAL= 1
            VALUE= "            "
      END_OBJECT= VERSIONID

      GROUP= INFORMATIONCONTENT
            OBJECT= INFORMATIONCONTENTCONTAINER
            CLASS= "1"

                  OBJECT= PARAMETERNAME
                              CLASS = "1"
                              NUM_VAL = 1
                              VALUE = "LocalGranuleID"
                  END_OBJECT = PARAMETERNAME

                  OBJECT = PARAMETERVALUE
                              CLASS = "1"
                              NUM_VAL  = 1
                              VALUE = "         "
                  END_OBJECT = PARAMETERVALUE
```

Listing 11.2.3-1 - Target MCF Template for a Static Granule (cont.)

```
                  OBJECT = PARAMETERNAME
                              CLASS = "2"
                              NUM_VAL = 1
                              VALUE = "Science_Group"
                  END_OBJECT = PARAMETERNAME

                  OBJECT = PARAMETERVALUE
                              CLASS = "2"
                              NUM_VAL  = 1
                              VALUE = "         "
                  END_OBJECT = PARAMETERVALUE
            END_OBJECT = INFORMATIONCONTENTCONTAINER
      END_GROUP = INFORMATIONCONTENT
END_GROUP = INVENTORYMETADATA
END
```

Listing 11.2.3-2 - Sample Target MCF for a Static Granule

```
GROUP= INVENTORYMETADATA
      GROUPTYPE= MASTERGROUP

      OBJECT= SHORTNAME
            NUM_VAL= 1
            VALUE= " SYN_MISC "
      END_OBJECT= SHORTNAME

      OBJECT= VERSIONID
            NUM_VAL= 1
            VALUE= "1"
      END_OBJECT= VERSIONID

      GROUP= INFORMATIONCONTENT
            OBJECT= INFORMATIONCONTENTCONTAINER
            CLASS= "1"

                  OBJECT= PARAMETERNAME
                              CLASS = "1"
                              NUM_VAL = 1
                              VALUE = "LocalGranuleID"
                  END_OBJECT = PARAMETERNAME

                  OBJECT = PARAMETERVALUE
                              CLASS = "1"
                              NUM_VAL  = 1
                              VALUE = " CER11T.mcf"
                  END_OBJECT = PARAMETERVALUE
```

Listing 11.2.3-2 - Sample Target MCF for a Static Granule (cont.)

```
                  OBJECT = PARAMETERNAME
                              CLASS = "2"
                              NUM_VAL = 1
                              VALUE = "Science_Group"
                  END_OBJECT = PARAMETERNAME

                  OBJECT = PARAMETERVALUE
                              CLASS = "2"
                              NUM_VAL  = 1
                              VALUE = "M1"
                  END_OBJECT = PARAMETERVALUE
            END_OBJECT = INFORMATIONCONTENTCONTAINER
      END_GROUP = INFORMATIONCONTENT
END_GROUP = INVENTORYMETADATA
END
```

11.2.4 Inserting Static Data Granules to the IMF Data Server

In order for static data files to be used both during the SSI&T and in production, this file must exist in the IMF Data Server and be accessible by the local machine. A program called the Insert Static File can be used for Inserting a static data granule into the IMF Data Server. Note that Source MCFs are static granules which must be inserted in this way.

The Activity Checklist table that follows provides an overview of the process for Inserting a static data granule into the IMF Data Server. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.2.4-1 Inserting Static Data Granules to the IMF Data Server - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the Insert Static File tool.	(I) 11.2.4	
2	SSI&T	Specify configuration file name.	(I) 11.2.4	
3	SSI&T	Specify mode.	(I) 11.2.4	
4	SSI&T	Specify ESDT ShortName.	(I) 11.2.4	
5	SSI&T	Specify type of static file.	(I) 11.2.4	
6	SSI&T	Specify PGE name.	(I) 11.2.4	
7	SSI&T	Specify PGE version.	(I) 11.2.4	
8	SSI&T	Specify file name of granule to Insert.	(I) 11.2.4	
9	SSI&T	Specify Target MCF file name.	(I) 11.2.4	
10	SSI&T	Specify quit or continue with other Inserts.	(I) 11.2.4	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager is running.
2. The ESDT descriptor file for the granule to be Inserted exists in the ODL format on the IMF Data Server.
3. The Target MCF for this data granule has been created for the Insert.

To Insert the static granule data file to the IMF Data Server, execute the steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Data Server** and then **Insert Static**.

- An xterm in which DpAtInsertStaticFile.sh is running will be displayed.
- Alternatively, the same tool can be invoked by typing at a UNIX prompt on an AIT Sun **DpAtInsertStaticFile.sh**, press **Return**.
- The DpAtInsertStaticFile.sh program will prompt for further information.

2 At the program prompt **Config filename (default *defaultConfigFile*)?**, press **Return**.

- The default configuration file for this tool will be used.
- The ***defaultConfigFile*** will be replaced by the full path name and file name of the default configuration file. The file name will be DpAtIS_*daac*.CFG where *daac* will be replaced by one of {GSFC, EDC, LARC, NSIDC}.

3 At the program prompt **mode (default ops)?**, press **Return**

- If **ops** is not the correct mode, then enter in the correct mode (*e.g.* ssit) and press **Return**.

4 At the program prompt **ESDT name?** type ***ESDTShortName***, press **Return**

- The ***ESDTShortName*** is the ShortName of the ESDT descriptor file corresponding to this granule to be Inserted. For example, type **CADMERLW**, press **Return**.

5 At the program prompt **Science group?**, type ***ScienceGroupID***, press **Return**

- The ***ScienceGroupID*** is an identifier used to define the file type as a coefficient file, a lookup table file,, or a MCF. It distinguishes static granules of different types which share the same ESDT. For instance, for a coefficient file, use **C*n***, where number *n* could be 0, 1, 2…; this number *n* needs to be matched with the number *n* in the PGE_PGEName#Version.odl file (see Section 11.1.2). For example, type **C1**, press **Return**.
- The Science Group ID must match what was edited into the PGE metadata ODL file for that PCF entry (Section 11.1.2).

6 At the program prompt **PGE name?**, type ***PGEName***, press **Return**.

- The ***PGEName*** is the name of the PGE for which this static granule is being Inserted. For example, type **PGE1aT**, press **Return**.
- The ***PGEName*** must match exactly the PGE name entered into the PDPS for this PGE (see Section 11.1.2).

7 At the program prompt **PGE version (default 1)?**, type ***PGEVersion***, press **Return**.

- The ***PGEVersion*** is the version of the PGE entered in step 6 for which this static granule is being Inserted. For example, type **1**, press **Return**.
- The ***PGEVersion*** must match exactly the PGE version entered into the PDPS for this PGE (see Section 11.1.2).

8 At the program prompt **Filename to insert?**, type ***pathname/GranuleFileName***, press **Return**

- The ***pathname*****/*GranuleFileName*** is the full path name and file name of the static data granule to be Inserted. For example, type **/data/CERES/CADMERLW_1**, press **Return**.

9 At the program prompt **Associated ASCII metadata (target MCF) filename to insert (default *pathname*/*GranuleFileName*.met)?**, press **Return**.
- The default is the file name of the granule to insert with the .met file name extension. If the default is not correct, then the file name of this file must be entered.
- The static data granule will be Inserted to the IMF Data Server.

10 At the program prompt **Hit return to run again, 'q <return>' to quit:** type **q** and press **Return** to quit or just press **Return** to insert additional dynamic granules.
- If continuing, repeat steps 2 through 9.

Table 11.2.4-2 Inserting Static Data Granules to the IMF Data Server - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Data Server → Insert Static	(No action)
2	(No entry)	press Return
3	(No entry)	press Return
4	*ESDTShortName*	press Return
5	*ScienceGroupID*	press Return
6	*PGEName*,	press Return
7	*PGEVersion*	press Return
8	*pathname*/*GranuleFileName*	press Return
9	(No entry)	press Return
10	q | Return	press Return

11.3 Placing the Science Software Executable on the IMF Data Server

11.3.1 Assembling a Science Software Executable Package

This section describes how to assemble a Science Software executables Package (SSEP) and create a corresponding Target MCF. A SSEP is a UNIX tar file which contains PGE executables, SDP Toolkit message files, and a Bourne shell profile, if applicable.

A Bourne shell profile file contains any external environment variables needed by the PGE at runtime. The file **must** be named **profile**.

For example:

```
BRAND=sgi

export BRAND
```

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/ecs/Rel_A/COTS/imsl/vn
i/lib/lib.sgi:/usr/ecs/Rel_A/COTS/sybase/lib

export LD_LIBRARY_PATH
```

In order to Insert a SSEP into the IMF Data Server (Section 11.3.2), a corresponding Target MCF must be generated for the SSEP. Such an ASCII metadata ODL file can be obtained by editing an existing template ODL file with the information of the specific PGE. The following procedures describe how to assemble a SSEP and create an ASCII metadata ODL file.

The Activity Checklist table that follows provides an overview of the process for assembling a SSEP and creating a corresponding ASCII metadata ODL file. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.3.1-1 Assembling a Science Software Executable Package - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Make a new directory to contain the contents of the SSPE.	(I) 11.3.1	
2	SSI&T	Change directories to this new directory.	(I) 11.3.1	
3	SSI&T	Copy all files to be placed into the SSEP into the new directory.	(I) 11.3.1	
4	SSI&T	Invoke UNIX command *tar* specifying PGE executables, message files, and Bourne shell profile (if necessary).	(I) 11.3.1	
5	SSI&T	Copy over a Target MCF template.	(I) 11.3.1	
6	SSI&T	Invoke editor to edit the Target MCF.	(I) 11.3.1	
7	SSI&T	Save the changes and exit the editor.	(I) 11.3.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. PGE executables, message files, and the Bourne shell profile (if necessary) are available to make a SSEP.

To create a SSEP, execute the steps that follow:

1 At the UNIX prompt on an AIT Sun, type **mkdir *SSEPpathname***, press **Return**.

- The ***SSEPpathname*** is the full path name of a *new* directory which will contain all the files to be placed into the SSEP as well as the SSEP itself.
- It is recommended that ***SSEPpathame*** be named with a convention that indicates the PGE for which a SSEP will be created. For example, type **mkdir MOD35.ssep**, press **Return**.

2 At the UNIX prompt on the AIT Sun, type **cd *SSEPpathname***, press **Return**.

- The ***SSEPpathname*** is the directory name of the new directory created in step 1.

3 At the UNIX prompt on the AIT Sun, type **cp *pathname*/*file1 pathname*/*file2 … pathname*/*filen* .**, press **Return** (note the "dot" and then space at the end of the command).

- The ***pathname*/*file1, pathname*/*file2,…pathname*/*filen*** represents a list of path names and file names (delimited by spaces) to copy into the current directory, ***SSEPpathame*** (the "dot" represents the current directory and must be last in the command).
- For example, type **cp /data/MODIS/pge/MOD35.pge /data/MODIS/mcf/mod35.mcf /data/MODIS/MOD_13453 .**, press **Return** (note the space and then "dot" at the end of the command).
- The files copied into this directory should be the PGE executable, any shell scripts or other executables that are part of the PGE, SDP Toolkit message files, and the Bourne shell profile (if applicable).
- Files can be individually copied into the ***SSEPpathame*** directory. For example, type **cp /data/MODIS/pge/MOD35.pge .**, press **Return** (note the space and then "dot" at the end of the command). Repeat for each file needed in the SSEP for this PGE.

4 At the UNIX prompt on the AIT Sun, type **tar cvf *SSEPfilename*.tar ***, press **Return**.

- The ***SSEPfilename*.tar** is the file name for the SSEP tar file. The file name extension .tar is recommended but not required.
- The asterisk (*****) is a file name wildcard that represents all files in the current directory. This will place all files in the SSEP tar file.
- Once created, the contents of the SSEP tar file can be viewed by typing **tar tvf *SSEPfilename*.tar**, press **Return**.
- Do not apply compression (*e.g.* UNIX compress or gzip) to the tar file.

5 At the UNIX prompt on the AIT Sun, type **cp *filename.met.tpl filename.met,*** press **Return**.

- The ***filename.met.tpl*** is the file name of the template Target MCF for this SSEP. If a template is not available, see Appendix D or use one used for another SSEP.
- The ***filename.met*** is the file name of the Target MCF to be tailored for this SSEP.

6 At the UNIX prompt on the AIT Sun, type **vi *filename.met***, press **Return**.

- The ***filename.met*** is the Target MCF for this SSEP.
- This command invokes the *vi* editor. Edit the ***filename*.met** with the specific information for the SSEP to be inserted.

- The following guidelines should be followed when editing on the Target MCF (***filename*.met**):
  - The value for the VERSIONID object should be filled out with the proper PGE version. For example: “1” .
  - In the INFORMATIONCONTENTCONTAINER object,
    - The value for the PARAMETERNAME object of the class “1” should be filled out with the PGE name. For example: “BTS”.
    - The value for the PARAMETERNAME object of the class “2” should be filled out with the PGE Science Software Version. For example: “1”.
    - The value for the PARAMETERNAME object of the class “3” should be filled out with the Platform Name. For example: “IRIX”.
    - The value for the PARAMETERNAME object of the class “4” should be filled out with the Platform Version. For example: “6.2”.
    - The value for the PARAMETERNAME object of the class “5” should be filled out with the date to perform the Insertion. For example: “970319”.
    - The value for the PARAMETERNAME object of the class “6” should be filled out with the time to perform the Insertion. For example: “14:45:00”.
    -
  - A template Target MCF is shown in listing 9.4.1-1.

7 Save the changes made to the SSEP’s Target MCF (***filename*.met**) and exit the editor.
- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor’s documentation.

Table 11.3.1-2 Assembling a Science Software Executable Package- Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	mkdir *SSEPpathname*	press Return
2	cd *SSEPpathname*	press Return
3	cp *pathname*/*file1* *pathname*/*file2* … *pathname*/*filen* .	press Return
4	tar cvf *SSEPfilename*	press Return
5	cp *PGEName#vers.tpl* *filename*.met	press Return
6	vi *filename*.met	edit specific values for VALUE= in the filename.met if they are blank.
7	:wq	press Return

Listing 11.3.1-1 - Template Target MCF for a SSEP

```
GROUP = INVENTORYMETADATA
  GROUPTYPE = MASTERGROUP

  OBJECT = SHORTNAME
    NUM_VAL = 1
    VALUE  = "PGEEXE"
  END_OBJECT = SHORTNAME

  OBJECT = VERSIONID
    NUM_VAL = 1
    VALUE =                         /* Version of PGE */
  END_OBJECT = VERSIONID

  GROUP = INFORMATIONCONTENT

    OBJECT = INFORMATIONCONTENTCONTAINER
      CLASS  = "1"

      OBJECT = PARAMETERNAME
        CLASS  = "1"
        NUM_VAL = 1
        VALUE  = "ExePGEName"
      END_OBJECT = PARAMETERNAME
```

Listing 11.3.1-1 - Template Target MCF for a SSEP (cont.)

```
OBJECT = PARAMETERVALUE
  CLASS  = "1"
  NUM_VAL = 1
  VALUE =                     /* Name of PGE  */
END_OBJECT = PARAMETERVALUE

OBJECT = PARAMETERNAME
  CLASS = "2"
  NUM_VAL = 1
  VALUE = "ExePGESSWVersion"
END_OBJECT = PARAMETERNAME

OBJECT = PARAMETERVALUE
  CLASS = "2"
  NUM_VAL = 1
  VALUE =                     /* Version of Science Software */
END_OBJECT = PARAMETERVALUE

OBJECT = PARAMETERNAME
  CLASS = "3"
  NUM_VAL = 1
  VALUE = "ExePlatformOS"
END_OBJECT = PARAMETERNAME
```

Listing 11.3.1-1 - Template Target MCF for a SSEP (cont.)

```
OBJECT = PARAMETERVALUE

  CLASS = "3"

  NUM_VAL = 1

  VALUE =                     /* Name of Platform */

END_OBJECT = PARAMETERVALUE


OBJECT = PARAMETERNAME

  CLASS = "4"

  NUM_VAL = 1

  VALUE = "ExePlatformOSVersion"

END_OBJECT = PARAMETERNAME


OBJECT = PARAMETERVALUE

  CLASS = "4"

  NUM_VAL = 1

  VALUE =                     /* Version of Platform */

END_OBJECT = PARAMETERVALUE


OBJECT = PARAMETERNAME

  CLASS = "5"

  NUM_VAL = 1

  VALUE = "ExeInsertDate"

END_OBJECT = PARAMETERNAME
```

Listing 11.3.1-1 - Template Target MCF for a SSEP (cont.)

```
      OBJECT = PARAMETERVALUE
        CLASS = "5"
        NUM_VAL = 1
        VALUE =                     /* Date to perform the Insertion */
      END_OBJECT = PARAMETERVALUE

      OBJECT = PARAMETERNAME
        CLASS = "6"
        NUM_VAL = 1
        VALUE = "ExeInsertTime"
      END_OBJECT = PARAMETERNAME

      OBJECT = PARAMETERVALUE
        CLASS = "6"
        NUM_VAL = 1
        VALUE =                     /* Time to perform the Insertion */
      END_OBJECT = PARAMETERVALUE

    END_OBJECT = INFORMATIONCONTENTCONTAINER

  END_GROUP = INFORMATIONCONTENT

END_GROUP = INVENTORYMETADATA

END
```

11.3.2 Inserting a Science Software Executable Package to the IMF Data Server

Science software, like any other data that are managed in the ECS, must be placed on the IMF Data Server. A program called the SSEP Insertion Tool can be used for Inserting a Science Software Executable Package into the IMF Data Server.

The Activity Checklist table that follows provides an overview of the process for Inserting a SSEP into the IMF Data Server. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 11.3.2-1 Inserting a Science Software Executable Package to the IMF Data Server - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the SSEP Insertion Tool.	(I) 11.3.2	
2	SSI&T	Specify configuration file name.	(I) 11.3.2	
3	SSI&T	Specify the mode.	(I) 11.3.2	
4	SSI&T	Specify PGE name.	(I) 11.3.2	
5	SSI&T	Specify software version.	(I) 11.3.2	
6	SSI&T	Specify file name of SSEP to Insert.	(I) 11.3.2	
7	SSI&T	Specify Target MCF file name.	(I) 11.3.2	
8	SSI&T	Specify top level shell within SSEP.	(I) 11.3.2	
9	SSI&T	Specify quit or continue with other Inserts.	(I) 11.3.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ESDT descriptor file for the SSEP to be Inserted exists in the ODL format on the IMF Data Server.
2. A Target MCF for this SSEP has been created for the Insert.
3. The SSEP has been created according to the procedures in Section 11.3.1.

To Insert the SSEP to the IMF Data Server, execute the steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Data Server** and then **Insert EXE TAR**.

- An xterm in which DpAtInsertExeTarFile.sh is running will be displayed.

- Alternatively, the same tool can be invoked by typing at a UNIX prompt on an AIT Sun **DpAtInsertExeTarFile.sh**, press **Return**.
- The DpAtInsertExeTarFile.sh program will prompt for further information.

2 At the program prompt **Config filename (default *defaultConfigFile*)?**, press **Return**.
- The default configuration file for this tool will be used.
- The ***defaultConfigFile*** will be replaced by the full path name and file name of the default configuration file. The file name will be DpAtIE_*daac*.CFG where *daac* will be replaced by one of {GSFC, EDC, LARC, NSIDC}.

3 At the program prompt **mode (default ops)?**, press **Return**
- If **ops** is not the correct mode, then enter in the correct mode (*e.g.* SSIT) and press **Return**.

4 At the program prompt **PGE name?**, type ***PGEName***, press **Return**.
- The ***PGEName*** is the name of the PGE for which this static granule is being Inserted. For example, type **PGE1aT**, press **Return**.
- The ***PGEName*** must match exactly the PGE name entered into the PDPS for this PGE (see Section 11.1.2).

5 At the program prompt **Science software version (default 1)?**, type ***SSWversion***, press **Return**.
- The ***SSWversion*** is the version of the science software which is being Inserted in this SSEP. Press **Return** to accept the default or enter in a version and press **Return**.

6 At the program prompt **Filename to insert?**, type ***pathname/SSEPFileName***, press **Return**
- The ***pathname/SSEPFileName*** is the full path name and file name of the SSEP tar file to be Inserted. For example, type **/data/MOD35/ssep/MOD35_1.tar**, press **Return**.
- The SSEP tar file must not be compressed (*e.g.* with UNIX compress or gzip).

7 At the program prompt **Associated ASCII metadata (target MCF) filename to insert (default *pathname*/*SSEPFileName*.met)?**, press **Return**.
- The default is the file name of the granule to insert with the .met file name extension. If the default is not correct, then the file name of this file must be entered.

8 At the program prompt **Top level shell filename within the tar file?**, type ***ExecFileName***, press **Return**.
- The ***ExecFileName*** is the file name of the top level executable within the SSEP tar file. It should be the same as was entered into the PDPS/SSIT Database Update GUI (Section 11.1.3).
- The SSEP will be Inserted to the IMF Data Server.

9 At the program prompt **Hit return to run again, 'q <return>' to quit:** type **q** and press **Return** to quit or just press **Return** to insert additional dynamic granules.

- If continuing, repeat steps 3 through 10.

Table 11.3.2-2 Inserting a Science Software Executable Package to the IMF Data Server - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Data Server → Insert EXE TAR	(No action)
2	(No entry)	press Return
3	(No entry)	press Return
4	*PGEName*	press Return
5	*SSWversion*	press Return
6	*pathname/SSEPFileName*	press Return
7	(No entry)	press Return
8	*ExecFileName*	press Return
9	q | Return	press Return

12. PGE Planning, Processing, and Product Retrieval

12.1 Registering Subscriptions for Test Data Files

Before a PGE can be run within the PDPS, subscriptions for the output data files, input data files, and the MCF(s) must be registered. This procedure describes how to register subscriptions.

The Activity Checklist table that follows provides an overview of the process for registering a subscription for test data files. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.1-1 Registering Subscriptions for Test Data Files - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the Subscription Editor from the SSIT Manager	(I) 12.3	
2	SSI&T	Register a Subscription for each of the Test Output Files	(I) 12.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The test data files have registered ESDTs (Section 5.3) with known ShortNames.

To register subscriptions for test data files, execute the steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Data Server** and then **Register Subscription**.
- An xterm in which PlSubsEdit.sh is running will be displayed.
- Alternatively, the same tool can be invoked by typing at a UNIX prompt on an PLN Sun **PlSubsEdit.sh**, press **Return**.
- The PlSubsEdit.sh program will prompt for further information.

2 At the program prompt **Config filename (default *defaultConfigFile*)?**, press **Return**.
- The default configuration file for this tool will be used.

- The ***defaultConfigFile*** will be replaced by the full path name and file name of the default configuration file. The file name will be PlSubsEdit_*daac*.CFG where *daac* will be replaced by one of {GSFC, EDC, LARC, NSIDC}.

3 At the program prompt **mode (default ops)?**, press **Return**
- If **ops** is not the correct mode, then enter in the correct mode (*e.g.* ssit) and press **Return**.

4 At the program prompt **Would you like to view the complete list of ESDTs known to PDPS? (y/n):**, type **y** and press **Return** to view the list of ESDTs or type **n** and press **Return** if the ESDT for which a subscription is being registered is known.
- If **y** is entered, a list of ESDTs known to the PDPS will be displayed.

5 At the program prompt **Is recipient PLS Subscription Manager (Y/N):**, type **Y** and press **Return**
- This prompt is due to “orphaned” Release A functionality.

6 At the program prompt **Enter ESDT short name:**, type ***ShortName*** and press **Return**
- This ***ShortName*** is the ESDT ShortName of the collection for which a granule subscription is being registered. This will be an input data file, the output data file, or the MCF(s) required by the PGE.

7 At the program prompt **Submit (S)/Withdraw (W):**, type **S** and press **Return**
- This results in the subscription being submitted.
- A message stating “Execution completed successfully” should be displayed.

8 At the program prompt **Hit return to run again, ‘q <return>’ to quit:** type **q** and press **Return** to quit or just press **Return** to register additional subscriptions.
- If continuing, repeat steps 3 through 7.

Table 12.1-2 Registering Subscriptions for Test Data Files - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Data Server → Register Subscription	(No action)
2	(No entry)	press Return
3	(No entry)	press Return
4	y | n	press Return
5	Y	press Return
6	*ShortName*	press Return
7	S	press Return
8	q | Return	press Return

12.2 Using the Production Request Editor

After a PGE has been successfully linked to the SCF Version of the SDP Toolkit, run from the command line, and then linked to the DAAC Version of the SDP Toolkit, the PGE is ready to be run through the automated ECS PDPS. A Production Request (PR) is then submitted for the science data processing job. The Production Request Editor accomplishes this function. Only one PR may be submitted at a time. A single PR is exploded by the PDPS into one or more jobs called Data Processing Requests (DPRs). The number of DPRs that are created for a single PR is determined by the number needed to cover the requested time interval. Some PRs may only require one DPR.

12.2.1 Invoking the Production Request Editor

The Production Request Editor is invoked from a PR Editor icon on the ECS Desktop. Clicking on the icon for the PR Editor brings up a screen with five tabs at the top for selection. The first tab is labeled “Planning”. Selection of this tab displays a list of four capabilities available for the PR Editor by selecting the other tabs at the top of the primary GUI screen: PR Edit, PR List, DPR View, and DPR List.

The Activity Checklist table that follows provides an overview of the process for invoking the Production Request Editor GUI. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.2.1-1 Invoking the Production Request Editor - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into a PLN Sun.	(I) 12.2.1	
2	SSI&T	Invoke the Production Request Editor.	(I) 12.2.1	
3	SSI&T	Choose a task.	(I) 12.2.1	
4	SSI&T	When tasks are completed, quit the Production Request Editor.	(I) 12.2.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PGE and all associated files have been registered in the PDPS Database.
2. All ESDTs required by the PGE have been registered.

3. The PGE has been successfully compiled and linked with the DAAC version of the SDP Toolkit.

To invoke the Production Request Editor GUI, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to a PLN Sun.
- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the PLN Sun.
- It is recommended that this procedure begin within a new command shell on a PLN Sun.

2 At the UNIX prompt on the PLN Sun, type **PlPREditor ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/PlPREditor_*daac*.CFG ecs_mode *mode* &**, press **Return**.
- The ***daac*** is one of {GSFC, EDC, LARC, NSIDC} .
- The ***mode*** is the operations mode.
- The Production Request Editor GUI will be displayed.
- For example, type **PlPREditor ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/PlPREditor_GSFC.CFG ecs_mode ops &**, press **Return**.
- Various messages from the Production Request Editor may appear in this window as it is running. For this reason, avoid using this window for other tasks until the Production Request Editor has terminated.

3 In the Production Request Editor, click on one of the tabs PR Edit, PR List, DPR View, or DPR List corresponding to desired task.
- To define a new Production Request or edit a Production Request, click on PR Edit. Proceed to Section 12.2.2.
- To review or list a Production Request, click on PR List.
- To view or inspect a Data Processing Request, click on DRP View.
- To review or inspect a Data Processing Request, click on DRP List.

4 When tasks are completed in the Production Request Editor GUI, click on the **File** menu, then choose **Exit**.
- The Production Request Editor will disappear.

Table 12.2.1-2 Invoking the Production Request Editor - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	**Tools → Xterm**	**telnet to PLN Sun**
2	**PlPREditor ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/PlPREditor_*daac*.CFG ecs_mode *mode***	**press Return**
3	(No entry)	**choose a Production Request activity**
4	**File → Exit**	(No action)

12.2.2 Defining a New Production Request

A Production Request (PR) is a request for data production of granules between a start date/time and an end date/time. A PR will explode into one or more Data Processing Requests (DPR) depending upon the time interval involved. Each DPR corresponds to the execution of a PGE. Therefore, a PR results in the execution of a PGE one or more times. Only one PGE is involved in a single Production Request.

This procedure describes how to define a new PR for a PGE that has already been registered in the PDPS database (see Section 11.1). It can also be used to modify an existing PR. It assumes that the **PR Edit** tab has been selected from the Production Request Editor.

The Activity Checklist table that follows provides an overview of the process for defining a new Production Request via the Production Request Editor. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.2.2-1 Defining a New Production Request - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke PR Edit page from Production Request Editor.	(I) 12.2.2	
2	SSI&T	Verify that the PR is indicated as being “New”.	(I) 12.2.2	
3	SSI&T	Select a PGE.	(I) 12.2.2	
4	SSI&T	Specify the priority of the PR.	(I) 12.2.2	
5	SSI&T	Optionally, view or override PGE parameter mappings.	(I) 12.2.2	
6	SSI&T	Specify start and end dates of the PR.	(I) 12.2.2	
7	SSI&T	Specify start and end times of the PR.	(I) 12.2.2	
8	SSI&T	Optionally, enter a comment for the PR.	(I) 12.2.2	
9	SSI&T	Save the PR under a PR name.	(I) 12.2.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The user has selected the **PR Edit** tab from the Production Request Editor (Section 12.1.1).
2. The PGE involved in the Production Request has been registered in the PDPS database (Section 11.1).

To define a new Production Request, execute the procedure steps that follow:

1 From the Production Request Editor GUI, click on the **PR Edit** tab.
- The PR Edit page will be displayed.

2 In the field labeled **PR Name:**, enter **New** or verify that **New** is already entered as the default.

3 The PGE for the Production Request must be selected from a list. To do this, click on the **PGE…** button.
- A GUI labeled **PGE Selection** will be displayed within which registered PGEs will be listed. The appropriate PGE can then be selected by clicking on it and then on the **OK** button.
- The selected PGE will then be used to populate **Satellite Name**, **Instrument Name**, **PGE Name**, and **PGE Version** fields of the main GUI.

4 In the field labeled **Priority:**, enter ***priority***.
- The ***priority*** is the priority to be assigned to this Production Request in the range 0 through 99 with 0 being the lowest priority and 99 the highest. For example, enter **40**.

5 Optionally, to view and possibly override PGE runtime parameter settings, click on the **PGE Parameters…** button.
- A GUI labeled **PGE Parameter Mappings** will be displayed.
- Parameter names, logical IDs (from the PCF), and default settings will be listed. To override a setting, click on a parameter name. In the field labeled **Parameter Mapping** enter a new value and press **Return**. The value entered will show up in the **Value Override** column.
- Other parameters may be changed in the same way.
- When complete, click on the **OK** button. The **PGE Parameter Mappings** GUI will disappear.

6 In the Production Request Editor GUI, enter ***StartDate*** and ***EndDate*** in fields labeled **Date:** next to the labels **Start:** and **End:**, respectively.
- The ***StartDate*** and ***EndDate*** are the start and end dates of the Production Request and should be entered in the mm/dd/yy format.

7 In the Production Request Editor GUI, enter ***StartTime*** and ***EndTime*** in fields labeled **Time:** next to the labels **Start:** and **End:**, respectively.
- The ***StartTime*** and ***EndTime*** are the start and end times of the Production Request and should be entered in the hh:mm:ss format.

8 Optionally, enter ***Comment*** in field labeled **Comment:**.
- This comment will be displayed whenever this Production Request is brought up and viewed.

9 When Production Request is complete, click on **File** menu and select **Save As…**.
- A GUI labeled **File Selection** will be displayed.

- In the field labeled **Selection**, enter a user-defined name to be assigned to the Production Request. Then click on the **OK** button. A message box will be displayed stating "Production Request Explosion into DPRs ok, *n* DPRs Generated", where *n* will be the number of DPRs. Click on the **Ok** button. A second message box stating "Write to Database of Production Request ok"; again click **Ok**.
- Note that you will not be allowed to enter a PR name that already exists. PR names that already exist will be displayed in the main window.
- The Production Request will then be saved under the name specified.

Table 12.2.2-2 Defining a New Production Request - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	**PR Edit**	**click tab**
2	**New**	**verify or enter**
3	**PGE…**	**click button then select a PGE**
4	***priority***	**enter**
5	(Optional) **PGE Parameters…**	**click button then set a parameter value**
6	***StartDate*** ***EndDate***	**enter** **enter**
7	***StartTime*** ***EndTime***	**enter** **enter**
8	(Optional) ***Comment***	**click parameter**
9	**File → Save As, then PR name**	**enter**

12.2.3 Viewing Production Requests

A Production Request (PR) is a request for data production of granules between a start date/time and an end date/time. A PR will explode into one or more Data Processing Requests (DPR) depending upon the time interval involved. Each DPR corresponds to the execution of a PGE. Therefore, a PR results in the execution of a PGE one or more times. Only one PGE is involved in a single Production Request.

This procedure describes how to view existing PRs that have already been defined (see Section 12.1.2). It assumes that the **PR List** tab has been selected from the Production Request Editor.

The information listed for each PR is:

- PR Name - The name assigned to the Production Request when it was defined.
- PGE ID - The name of the PGE involved in the PR.
- Priority - The priority (0 - 99) of the PR assigned when it was defined.
- Start - The start date and time of the PR.

- End - The end date and time of the PR.
- Comment - Any comment that was entered when the PR was defined.

The Activity Checklist table that follows provides an overview of the process for viewing existing Production Requests. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.2.3-1 Viewing Production Requests - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke PR List page from Production Request Editor.	(I) 12.2.3	
2	SSI&T	View the listed PRs.	(I) 12.2.3	
3	SSI&T	Optionally, save PR under new name and modify.	(I) 12.2.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The user has selected the **PR List** tab from the Production Request Editor (Section 12.2.1).

To view Production Requests, execute the procedure steps that follow:

1 From the Production Request Editor GUI, click on the **PR List** tab.
- The PR List page will be displayed.

2 View the listed PRs. Optionally, find a PR by entering a search string in the field next to the **Find** button and then clicking on the **Find** button.

3 To modify a PR listed, click on the PR in the list and from the **File** menu select **Save As…**.
- In the **File Selection** GUI, replace the current PR name shown in the **Selection** field with a new PR name. Then click on the **OK** button.
- When modifying an existing PR, it must be saved under a new PR name.
- Next, click on the **PR Edit** tab. The PR Edit page will be displayed with fields populated from the existing PR name, but having the new PR name chosen above.
- See Section 12.1.2 on using the PR Edit page and saving any changes made.

Table 12.1.3-2 Viewing Production Requests - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	PR List	click tab
2	(Optional) *search string*	click Find button
3	(Optional) *PGE*	click PGE

12.2.4 Viewing Data Processing Requests

Clicking on the DPR View GUI tab displays a list of all DPRs for all PRs entered into the system.

The Activity Checklist table that follows provides an overview of the process for invoking the DPR View GUI. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.2.4-1 Viewing Data Processing Requests - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	From the PR Editor GUI, select DPR View GUI tab.	(I) 12.1.5	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

To invoke the DPR View GUI, execute the procedure steps that follow:

1 From the PR Editor GUI, click on **DPR View** tab.

Table 12.2.4-2 Viewing Data Processing Requests - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	DPR View tab	Click on tab

12.2.5 Listing Data Processing Requests

Selection of one PR on the PR List by highlighting it and then clicking on the DPR List tab, brings up a detailed display of all DPRs associated with the selected PR. These may be examined in order to develop production plans and schedule jobs.

The Activity Checklist table that follows provides an overview of the process for invoking the DPR List GUI. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.2.5-1 Listing Data Processing Requests - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	From the PR Editor GUI, select DPR List GUI tab.	(I) 12.1.6	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

To invoke the DPR List GUI, execute the procedure steps that follow:

1 From the PR Editor GUI, click on **DPR List** tab**.**

Table 12.2.5-2 Listing Data Processing Requests - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	DPR List tab	Click on tab

12.3 Using the Production Planning Workbench

The Production Planner uses the Production Planning Workbench to create new production plans and display a planning timeline.

12.3.1 Using the Planning Workbench to Run One PGE

Once a PGE has been fully registered (Section 11.1), its test data files have been Inserted to the IMF Data Server (Section 11.2), and subscriptions for the test data have been registered (Section 12.1), the PGE can be run in an isolation test. An isolation test runs one execution of a single PGE. In such a test, the Planning Workbench is used to plan a Production Request (PR) such that a single Data Processing Request (DPR) is generated and run.

The Activity Checklist table that follows provides an overview of the process for using the Planning Workbench to run one PGE. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where

details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.3.1.1- Using the Planning Workbench to Run One PGE - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into a PLN Sun.	(I) 12.3.1	
2	SSI&T	Invoke Planning Workbench.	(I) 12.3.1	
3	SSI&T	Select a Production Request to plan.	(I) 12.3.1	
4	SSI&T	Schedule the Production Request.	(I) 12.3.1	
5	SSI&T	Activate the plan.	(I) 12.3.1	
6	SSI&T	Adjust the plan time.	(I) 12.3.1	
7	SSI&T	Exit the Planning Workbench.	(I) 12.3.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The required UNIX environment variables have been set properly.

To use the Planning Workbench to run one PGE, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to a PLN Sun.

- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the PLN Sun.
- It is recommended that this procedure begin within a new command shell on a PLN Sun.

2 At the UNIX prompt on an AIT Sun, type **/usr/ecs/Rel_A/CUSTOM/bin/PLS/st_all.*daac*.csh**, press **Return**.

- The ***daac*** is one of {gsfc, edc, larc, nsidc}. Note that, unlike in similar situations, the DAAC name is *lower* case.
- A GUI labeled **Message Handler** will be displayed.
- Shortly after, the Planning Workbench GUI will be displayed.

3 In the Planning Workbench GUI, go to the subwindow labeled **Unscheduled** and click on a Production Request name.

- The Production Request name is the name under which the PR was saved in Section 12.2.2.
- The PR name entry will be highlighted.

4 In the Planning Workbench GUI, click on the button next to the label **Schedule** (the button has an inverted triangle on it).

- The PR highlighted in step 3 will appear in the subwindow labeled **Scheduled**.

5 In the Planning Workbench GUI, click on the **Activate** button

- A small GUI labeled **Plan Activation** will be displayed.

6 In the Plan Activation GUI, set the time in the time field forward to allow ample time for the PGE to run. Then click on the **Ok** button.

- All that is necessary is for there to be sufficient time for the PGE run. There is no penalty for allowing *too* much time.
- The Production Request thus planned will be submitted to processing and its progress can be monitored with AutoSys (see Section 12.4).

7 When tasks are completed with the Planning Workbench GUI, click on the **File** menu, then choose **Exit**.

- The Planning Workbench GUI will disappear.

Table 12.3.1-2 Using the Planning Workbench to Run One PGE - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to PLN Sun
2	/usr/ecs/Rel_A/CUSTOM/bin/PLS/st_all.*daac*.csh	press Return
3	*PR name*	click PR name
4	Schedule	click button
5	Activate	click button
6	*time*	adjust time and click Ok
7	File → Exit	(No action)

12.4 Monitoring Production

The progress of one or more PGEs running within the PDPS may be monitored. The COTS tool used for this purpose is AutoSys® by Atria Software.

Each Data Processing Request results in seven AutoSys jobs that are boxed together. An AutoSys job name follows the template:

PGEname#versionMMDDYYhhmm.Suffix

where *PGEname* is replaced by the name of the PGE; *version* is replaced by the version of the PGE; *MMDDYY* is replaced by the two-digit month (*MM*), day (*DD*), and year (*YY*) that the job is scheduled to run; *hhmm* is replaced by the time (hours and minutes) that the job is scheduled to run; and *Suffix* is a two character extension indicating the job phase of the DPR. Refer to Table 12.4-1.

For example, for version 2 of a PGE named MOPITT4 scheduled to run on August 18, 1998 at 1:00pm, the AutoSys jobs making up that DPR would be:

MOPITT4#20818981300.Al

MOPITT4#20818981300.St

MOPITT4#20818981300.Pr

MOPITT4#20818981300.EX

MOPITT4#20818981300.Ps

MOPITT4#20818981300.Ds

MOPITT4#20818981300.Da

Table 12.4-1 AutoSys Jobs for a DPR - Activity Checklist

Job Name Suffix	Description
.Al	Resource allocation
.St	Staging
.Pr	Pre-processing
.EX	Execution of the PGE itself
.Ps	Post-processing
.Ds	De-staging
.Da	De-allocation of resources

The Activity Checklist table that follows provides an overview of the process for monitoring production. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.4-2 Monitoring Production - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the AutoSys monitor from the SSIT Manager	(I) 12.4	
2	SSI&T	Select the DPRs that are to be displayed in the ‘AutoSys Job Activity Console’ window.	(I) 12.4	
3	SSI&T	View details of a single DPR in the PDPS	(I) 12.4	

Table 12.4-2 Monitoring Production - Activity Checklist (cont.)

Order	Role	Task	Section	Complete?
4	SSI&T	View the existing Event Report on the selected DPR	(I) 12.4	
5	SSI&T	As necessary, modify the status of a DPR	(I) 12.4	
6	SSI&T	View or modify the details of any processing alarms for a DPR	(I) 12.4	
7	SSI&T	View the DPR job(s) upon which the selected DPR depends	(I) 12.4	
8	SSI&T	View the DPR job(s) that depend on the selected DPR	(I) 12.4	
9	SSI&T	Exit the AutoSys monitor	(I) 12.4	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The required UNIX environment variables have been set properly. In particular, the environment variables AUTOSYS, AUTOUSER, and AUTOSERV must be set.

To monitor production, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to a PLN Sun.
- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the PLN Sun.
- It is recommended that this procedure begin within a new command shell on a PLN Sun.

2 At the UNIX prompt on the PLN Sun, type **autosc &**, press **Return**.
- A GUI labeled **AutoSys** will be displayed.
- This GUI will contain eight buttons for invoking various tools available under AutoSys.

3 In the AutoSys GUI, click on the **Ops Console** button.
- A GUI labeled **AutoSys Job Activity Console** GUI will be displayed.
- The main subwindow of this GUI will contain a dynamically updated list of AutoSys jobs (seven jobs make up one DPR) currently scheduled.
- To disable dynamic updating of the main subwindow (which may be distracting), click on **Freeze Frame** in the small subwindow labeled **Show**.
- DPRs will be listed in the column labeled **Job Name** and their statuses (SUCCESS, FAILURE, TERMINATED) will be listed in the column labeled **Status**.

- To view job status information for a particular DPR, click on a DPR. Below the main subwindow, available job status information will be displayed.
- To view the existing event report, under the label **Reports**, click on the middle diamond labeled **Event**. The current event status for the selected DPR will be displayed in the subwindow labeled **Event Report**.
- Exit the **AutoSys Job Activity Console** by clicking on the **Exit** button.

4 In the AutoSys GUI, click on the **TimeScape** button.
- A GUI labeled **TimeScape** GUI will be displayed.
- The main subwindow labeled **Job Name** of this GUI will contain a dynamically updated list of AutoSys jobs (seven per DPR) currently scheduled.
- To disable dynamic updating of AutoSys jobs (which may be distracting), click on the **Freeze Frame** button.
- The color of each job indicates its status according to the legend on the left side of the GUI.
- The time line is shown on the right side of the GUI with time marked at the top. A red vertical line (dashed) indicates the current time.
- Exit the **TimeScape** GUI by clicking on the **File** menu and selecting **Exit**.

5 To quit AutoSys, in the AutoSys GUI, click on the **Exit** button.
- The AutoSys GUI will disappear.

Table 12.4-3 Monitoring Production - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to PLN Sun
2	autosc &	press Return
3	Ops Console	click Ops Console button
4	TimeScape	click TimeScape button
5	Exit	click Exit button

12.5 Using the Q/A Monitor

The Q/A Monitor allows the output products produced during a PGE run to be accessed and examined. Input test data granules and Production History files can be retrieved in the same manner. The Q/A Monitor retrieves output products based on the collection name (*i.e.* the ESDT) and time of Insertion to the IMF Data Server.

The Activity Checklist table that follows provides an overview of the process for using the Q/A Monitor. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 12.5-1 Using the Q/A Monitor - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log into a PLN Sun.	(I) 12.5	
2	SSI&T	Invoke the Q/A Monitor.	(I) 12.5	
3	SSI&T	Select the ESDT for which granules will be queried.	(I) 12.5	
4	SSI&T	Specify the date range over which to conduct the query.	(I) 12.5	
5	SSI&T	Submit the query.	(I) 12.5	
6	SSI&T	Select a data granule from the results list.	(I) 12.5	
7	SSI&T	Optionally, retrieve the Production History file for the selected granule.	(I) 12.5	
8	SSI&T	Retrieve the data granule.	(I) 12.5	
9	SSI&T	Choose the visualize data option.	(I) 12.5	
10	SSI&T	Select the file name corresponding to the granule selected.	(I) 12.5	
11	SSI&T	Invoke EOSView for the selected granule.	(I) 12.5	
12	SSI&T	When finished, exit the Q/A Monitor.	(I) 12.5	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The required UNIX environment variables have been set properly.
2. The desired output products have been successfully Inserted to the IMF Data Server.

To use the Q/A Monitor, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to a PLN Sun.
- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the PLN Sun.
- It is recommended that this procedure begin within a new command shell on a PLN Sun.

2 At the UNIX prompt on the PLN Sun, type **DpPrQaMonitorGUI ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/DpPrQaMonitorGUI_*daac*.CFG ecs_mode *mode***, press **Return**.
- The ***daac*** is one of {GSFC, EDC, LARC, NSIDC} .
- The ***mode*** is the operations mode.
- The Q/A Monitor GUI will be displayed.

- For example, type **DpPrQaMonitorGUI ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/DpPrQaMonitorGUI_EDC.CFG ecs_mode ops**, press **Return**.
- Various messages from the Q/A Monitor will appear in this window as it is running.

3 In the Q/A Monitor subwindow labeled **Data Types,** select an ESDT from the list presented and click on it.
- Use the scroll bars if necessary to locate desired ESDT.
- Optionally, use the **Find** field and button to locate an ESDT.

4 In the Q/A Monitor, under the label **Data Granule Insert Date (mm/dd/yy)**, set the date range within which the search for granules of the ESDT selected will be conducted.
- The date range can be made arbitrarily large to select all granules of a particular collection (ESDT).
- The dates refer to date of granule Insert to the IMF Data Server.

5 In the Q/A Monitor, click on the **Query** button.
- The results of the query will be displayed in the bottom window, labeled **Data Granules**.
- All granules having the ESDT selected in step 3 and having Insert times within the date range specified in step 4 will be listed in this window.

6 In the Q/A Monitor subwindow labeled **Data Granules**, click on one of the data granules listed to be examined.
- The data granule selected will be highlighted.

7 To retrieve the data granule's Production History file, click on the **Retrieve Prod History** button.
- The Production History (PH) tar file corresponding to the selected data granule will be retrieved from the IMF Data Server and placed on the local machine (a PLN Sun) in the directory /var/tmp.
- The PH can then be moved or copied manually from the /var/tmp directory to a user working directory for examination (see Section 16.1.2).
- Only the PH file is retrieved with the **Retrieve Prod History** button.
- If only the PH is desired, exit this procedure. To retrieve the data granule itself, continue on to step 8.

8 To retrieve the data granule, click on the **Retrieve Data Granule** button and note its file name (listed in the entry for the granule; you may have to scroll over to the right to see it).
- The data granule selected will be retrieved from the IMF Data Server and placed on the local machine (a PLN Sun) in the directory /var/tmp.

- A granule of any format (binary, ASCII, HDF, HDF-EOS) may be retrieved in this manner. Only HDF and HDF-EOS granules, however, may be further visualized using EOSView as described in the next steps.

9 To examine the data granule, click on the **Visualize data** tab.
- The **Visualize data** page will be displayed.

10 In the main subwindow on the **Visualize data** page, locate the file name of the granule retrieved in step 8 and click on it.
- The item selected will be highlighted and will appear in the **Selection** subwindow below.

11 Click on the **Visualize** button.
- This action will invoke EOSView with the granule selected.
- The granule must be HDF or HDF-EOS format.
- See Sections 14.1 and 14.2 for use of EOSView.

12 When tasks are completed with the Q/A Monitor GUI, click on the **File** menu, then choose **Exit**.
- The Q/A Monitor GUI will disappear.

Table 12.5-2 Using the Q/A Monitor - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to PLN Sun
2	DpPrQaMonitorGUI ConfigFile /usr/ecs/Rel_A/CUSTOM/cfg/ DpPrQaMonitorGUI_*daac*.CFG ecs_mode *mode*	press Return
3	*data type*	click entry
4	*data granule insert date*	adjust dates
5	Query	click Query button
6	*data granules*	click entry
7	(Optional) Retrieve Prod History	click button
8	Retrieve Data Granule	click button
9	Visualize data	click tab
10	*granule file name*	click entry
11	Visualize	click button
12	File → Exit	(No action)

13. File Comparison

An important part of SSI&T is verifying that the output files produced at the DAAC are identical (within particular tolerances) to the test output files delivered with the DAPs. A successful comparison is a strong indication that the porting of the science software from the development facility at the SCF to the operational facility at the DAAC has not introduced any errors.

A number of file comparison tools are available during SSI&T via the SSIT Manager GUI or they can be invoked from the UNIX command line. Two tools are available for comparing HDF or HDF-EOS files, one tool for comparing ASCII files, and another tool for assisting in comparing binary files.

It is assumed that the Instrument Team has delivered test output files (produced at their SCF) with which to perform the comparison.

13.1 Using the GUI HDF File Comparison Tool

Most output files (products) from PGEs run in the DAAC will be in HDF-EOS format. The GUI File Comparison Tool allows comparison of two HDF or HDF-EOS files.

The Activity Checklist table that follows provides an overview of the process for using the GUI HDF File Comparison Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 13.1-1 Using the GUI HDF File Comparison Tool - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the HDF File Comparison Tool from the SSIT Manager	(I) 13.1	
2	SSI&T	Log into the SPR SGI.	(I) 9.5	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. Two HDF or HDF-EOS files exist with similar structures, or at least having a data set in common.

2. The SSIT Manager is running.

3. If either of the two HDF/HDF-EOS files is in a ClearCase VOB, a ClearCase view was set before the SSIT Manager was started.

To compare two HDF or HDF-EOS files, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Product Examination**, then **HDF**.

- The HDF File Comparison Tool GUI will be displayed.

2 In the HDF File Comparison Tool GUI, click on the **File 1** button.

- Read the *Systems Description* document and the *Operations Manual*. Both of these or their equivalent should be in the delivery.

Table 13.1-2 Using the GUI HDF File Comparison Tool - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	(No entry)	**read all documentation**
2	**Tools → Xterm**	**telnet to SPR SGI**

13.2 Using the hdiff HDF File Comparison Tool

Most output files (products) from PGEs run in the DAAC will be in HDF-EOS format. The *hdiff* File Comparison Tool is a text-oriented tool run from the command line. It allows comparison of two HDF or HDF-EOS files.

The Activity Checklist table that follows provides an overview of the process for using the *hdiff* HDF File Comparison Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 13.2-1 Using the hdiff HDF File Comparison Tool - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the hdiff HDF File Comparison Tool from the SSIT Manager	(I) 13.2	
2	SSI&T	Specify the first file to compare.	(I) 13.2	
3	SSI&T	Specify the second file to compare.	(I) 13.2	
4	SSI&T	Examine the differences (if any).	(I) 13.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. Two HDF or HDF-EOS files exist with similar structures, or at least having a data set in common.
2. The SSIT Manager is running.
3. If either of the two HDF/HDF-EOS files is in a ClearCase VOB, a ClearCase view was set before the SSIT Manager was started.

To compare two HDF or HDF-EOS files, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Product Examination**, then **File Comparison**, and then **HDF (hdiff)**.
- An xterm window running *hdiff* will be displayed.

2 In the xterm window at the prompt **Options? (-h for help)**, type in any desired options and then press **Return**.
- To see the list of available options, type **-h** and press **Return** to the prompt.

3 In xterm window at the prompt **1st file to compare?**, type ***filename1***, press **Return.**
- The ***filename1*** is the file name of the first of two HDF or HDF-EOS files to be compared.
- If ***filename1*** is not in the current directory (the directory from which the SSIT Manage was run), include the full path name with the file name.

4 In xterm window at the prompt **2nd file to compare?**, type ***filename2***, press **Return.**
- The ***filename2*** is the file name of the second of two HDF or HDF-EOS files to be compared.
- If ***filename2*** is not in the current directory (the directory from which the SSIT Manage was run), include the full path name with the file name.
- The two files will be compared and the output will be displayed in the xterm window.

Table 13.2-2 Using the hdiff HDF File Comparison Tool - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Product Examination → File Comparison → HDF (hdiff)	(No action)
2	*filename1*	press Return
3	*filename2*	press Return
4	(No entry)	visually compare the two files

13.3 Using the ASCII File Comparison Tool

Most output files (products) from PGEs run in the DAAC will be in HDF-EOS format. A small minority may be in ASCII (text) format. The ASCII File Comparison Tool is a front-end to *xdiff* UNIX X Window tool for comparing two ASCII files.

The Activity Checklist table that follows provides an overview of the process for using the ASCII File Comparison Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 13.3-1 Using the ASCII File Comparison Tool - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the ASCII File Comparison tool from the SSIT Manager	(I) 13.3	
2	SSI&T	Specify the first file to compare.	(I) 13.3	
3	SSI&T	Specify the second file to compare.	(I) 13.3	
4	SSI&T	Examine the differences (if any).	(I) 13.3	
5	SSI&T	Quit the tool or perform another comparison.	(I) 13.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. Two ASCII files exist and have read permissions.
2. If either of the two ASCII files are in a ClearCase VOB, a ClearCase view was set before the SSIT Manager was started.

To compare two ASCII files, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Product Examination**, then **File Comparison**, and then **ASCII**.
- An xterm window running *xdiff* will be displayed.

2 In xterm window at the prompt **1st file to compare?**, type ***filename1***, press **Return.**
- The ***filename1*** is the file name of the first of two ASCII files to be compared.
- If ***filename1*** is not in the current directory (the directory from which the SSIT Manage was run), include the full path name with the file name.

3 In xterm window at the prompt **2nd file to compare?**, type ***filename2***, press **Return.**

- The ***filename2*** is the file name of the second of two ASCII files to be compared.
- If ***filename2*** is not in the current directory (the directory from which the SSIT Manage was run), include the full path name with the file name.
- A GUI labeled **xdiff** will be displayed.

4 In the GUI labeled **xdiff**, view the differences between the two files displayed.
- File ***filename1*** will be displayed on the left side of the GUI. File ***filename2*** will be displayed on the right.
- Only sections of file in which there are differences will be displayed. A "bang" character (!) at the beginning of a line indicates that a difference was found.
- For further help on *xdiff*, type **man xdiff**, press **Return** in an xterm window.
- Close the display window by using the pull down menu from the X window in the upper left corner.

5 In the xterm window at the prompt **Hit return for another diff, 'q <return>' to quit:**, type **q** and then press **Return** to quit or just press **Return** to perform another comparison.

Table 13.3-2 Using the ASCII File Comparison Tool - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Product Examination → File Comparison → ASCII	(No action)
2	*filename1*	press Return
3	*filename2*	press Return
4	(No entry)	visually compare the two files
5	q | Return	press Return

13.4 Using the Binary File Difference Assistant

Most output files (products) from PGEs run in the DAAC will be in HDF-EOS format. A small minority may be in some binary format. The Binary File Difference Assistant aids the user in constructing code that allows comparison of binary output files. Since there is an unwieldy number of possibilities for binary file formats, this tool cannot compare two binary files without some custom code written at the DAAC, hence, the "Assistant" in the name. The Binary File Difference Assistant aids the user by generating a makefile, a driver module, and a template comparison module in C, FORTRAN 77 or IDL (Interactive Data Language). The user then edits these templates to read the particular binary format in question according to a SCF-supplied format specification.

The Activity Checklist table that follows provides an overview of the process for using the Binary File Difference Assistant Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where

details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 13.4-1 Using the Binary File Difference Assistant - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the Binary File Difference Assistant.	(I) 13.4	
2	SSI&T	Select a programming language.	(I) 13.4	
3	SSI&T	Optionally, examine the example compare functions.	(I) 13.4	
4	SSI&T	Optionally, examine the example drivers.	(I) 13.4	
5	SSI&T	Optionally, examine the help display.	(I) 13.4	
6	SSI&T	Create the source code module, driver, and makefile.	(I) 13.4	
7	SSI&T	Customize the source code module, driver, and makefile. Build the executable and then run the program.	(I) 13.4	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. Two binary files exist.
2. Information on the data structure and tolerances have been provided.

To compare two binary files, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Product Examination**, then **File Comparison**, and then **Binary**.
- The Binary File Difference Assistant tool GUI will be displayed.

2 In the Binary File Difference Assistant tool GUI, click on one of the languages listed under the **Select Language** label. The choices are C, FORTRAN, or IDL.
- The choice of language depends largely on preference. It does not necessarily have to be the language that was used to create the files being compared.

3 Optionally, click on either the **Image** button or the **Structure** button located under the label **Compare Function**.
- Clicking on the **Image** button will display a code example for comparing binary files containing images.
- Clicking on the **Structure** button will display a code example for comparing binary files containing structures or records.
- The displayed listing well documented and should be read.
- The language of the code will depend on the language selection made in step 2.

4 Optionally, click on either the **Image** button or the **Structure** button located under the label **Driver**.

- Clicking on the **Image** button will display a code example for a driver invoking the compare function for binary files containing images.
- Clicking on the **Structure** button will display a code example for a driver invoking the compare function for binary files containing structures or records.
- The displayed listing well documented and should be read.
- The language of the code will depend on the language selection made in step 2.

5 Optionally, click on either the **Help** button.

- A Help GUI will be displayed.
- To end help, click on the **Dismiss** button.
- The Help GUI may remain displayed while using the Binary File Difference Assistant.

6 Once familiar with the code examples (steps 3 and 4), click on the **Copy** button.

- A GUI labeled **Enter Unique ID** will be displayed.
- In the field labeled **Enter unique file identifier:**, type ***fileID***, click on the **OK** button.
- The ***fileID*** will be used in the file names of the files copied over. These files will be:
  - **C:**
    - DaacBinDiff_***fileID***.c Compare function
    - DaacBinDiff_***fileID***_driver.c Driver
    - DaacBinDiff_***fileID***.mak Makefile
  - **FORTRAN:**
    - DaacBinDiff_***fileID***.f Compare function
    - DaacBinDiff_***fileID***_driver.f Driver
    - DaacBinDiff_***fileID***.mak Makefile
  - **IDL:**
    - DaacBinDiff_***fileID***.pro Compare function
    - DaacBinDiff_***fileID***_driver.pro Driver
    - DaacBinDiff_***fileID***.sh Shell script with here document
- The files will be copied into the directory from which the SSIT Manager is being run.

7 Using any desired text editor, customize the files for the job at hand. Then build the executable using the customized makefile provided (for C and FORTRAN). Then run the program to perform the binary file comparison.

Table 13.4-2 Using the Binary File Difference Assistant - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Product Examination → File Comparison → Binary	(No action)
2	language	click

Table 13.4-2 Using the Binary File Difference Assistant - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
3	(Optional) Compare Function → Image | Structure	click
4	(Optional) Driver → Image | Structure	click
5	(Optional) Help	click
6	(No entry)	customize files

14. Data Visualization

In order to fully ascertain the success of science software in producing scientifically valid data sets, the data need to be displayed in forms that convey the most information easily. Data visualization enables this to be done. This section describes the data visualization tools available at the DAACs during SSI&T and how to use them for detailed inspection of data sets produced by PGEs. Only some aspects of data visualization will be addressed in these procedures. For further information, see the references below.

The two visualization tools provided at the DAACs are EOSView and IDL. EOSView was developed by ECS and is a user-friendly GUI for creating two-dimensional displays from HDF-EOS objects (Grid, Swath) as well as the standard HDF objects (SDS, Vdata, Image, Text). It has additional features such as thumbnail-panning, colorization, zooming, plotting, and animation.

IDL (Interactive Data Language) is a COTS display and analysis tool widely applied in the scientific community. It is used to create two-dimensional, three-dimensional (volumetric), and surface/terrain displays from binary, ASCII, and many other CSDTs (or formats) in addition to HDF. IDL has additional features including flexible input/output, custom colorization, plotting, scripting, raster-math, geographic registration, map projection transformation, and vector map overlay.

14.1 Viewing Product Metadata with the EOSView Tool

This procedure describes how to use the EOSView tool to inspect the metadata in the HDF-EOS output file from a PGE. See Section 14.2 for how to inspect the science data. An HDF-EOS file is defined a file produced using the SDP Toolkit metadata tools and having, at a minimum, the ECS core metadata.

The Activity Checklist table that follows provides an overview of the process for viewing product metadata with the EOSView Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.1-1 Viewing Product Metadata with the EOSView Tool - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the EOSView GUI.	(I) 14.1	
2	SSI&T	Invoke the Open GUI.	(I) 14.1	
3	SSI&T	Select the directory containing the HDF-EOS file(s) to examine.	(I) 14.1	
4	SSI&T	Select the HDF-EOS file to examine.	(I) 14.1	
5	SSI&T	Select the HDF object in the file to examine.	(I) 14.1	
6	SSI&T	Display the object's global metadata.	(I) 14.1	
7	SSI&T	Repeat steps 5 and 6 for each object to examine.	(I) 14.1	
8	SSI&T	Close the HDF-EOS file.	(I) 14.1	
9	SSI&T	Quit the EOSView GUI.	(I) 14.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools.
2. EOSView has been properly installed and is accessible to the user.

To view product metadata with the EOSView tool, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Product Examination**, then **EOSView**.

- The EOSView GUI will be displayed.

2 In the GUI labeled **EOSView - EOSView Main Window**, click on the **File** menu and select **Open.**

- The **Filter** GUI will be displayed.

3 In the subwindow labeled **Filter**, enter full path name and file name wildcard template. For example, enter ***/home/MyDirectory/MySubdirectory/****.

- The ***/home/MyDirectory/MySubdirectory/**** represents the location to the directory containing the HDF-EOS files to examine.
- The asterisk (*) is a wildcard template that represents all files in that directory; other wildcard templates can narrow the search further, *e.g.* ***.hdf**.
- Use the **Directories** field to further select the correct directory.
- Files found matching the wildcard template in the chosen directory will be displayed in **Files** subwindow.

4 In the **Files** subwindow, click on the file name of the HDF-EOS file to examine. Then click on the **OK** button.

- A GUI labeled **EOSView - *MyOutputFile.hdf*** will be displayed where *MyOutputFile.hdf* is the file name of the file chosen in step 3.
- Be patient - this GUI may take some time to appear, particularly for large files.
- Once displayed, a list of HDF objects will appear in the main window. If nothing is listed, it means that no HDF objects were found within the file.

5 In the GUI labeled **EOSView - *MyOutputFile.hdf***, click on an object listed for which metadata is to be inspected.
- The object selected will be highlighted.
- Do not double click on object since this will cause a **Dimension** GUI to be displayed instead.

6 In the GUI labeled **EOSView - *MyOutputFile.hdf***, click on the **Attributes** menu and select **Global**.
- A GUI labeled **EOSView - Text Display** will be displayed.
- The global metadata associated with the object selected (in step 5) will be displayed in a scrollable field.
- If instead, the message "Contains no Global Attributes" appears, then the selected object contains no global metadata.

7 Repeat steps 5 and 6 for each HDF object within the selected HDF-EOS file for which metadata is to be examined.

8 In the GUI labeled **EOSView - *MyOutputFile.hdf***, click on the **File** menu and select **Close**.
- The **EOSView - *MyOutputFile.hdf*** GUI will disappear.
- Be patient - this GUI may take some time to disappear, particularly for large files.

9 In the GUI labeled **EOSView - EOSView Main Window**, click on the **File** menu and select **Exit**.
- The **EOSView - EOSView Main Window** GUI will disappear.

Table 14.1-2 Viewing Product Metadata with the EOSView Tool - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Product Examination → EOSView	(No action)
2	File → Open	(No action)
3	*/home/MyDirectory/MySubdirectory/**	select file filter
4	OK	select file
5	select object	click
6	Attributes → Global	(No action)
7	(No entry)	repeat 5 and 6
8	File → Close	(No action)
9	File → Exit	(No action)

14.2 Viewing Product Data with the EOSView Tool

This procedure describes how to use the EOSView tool to inspect the science data in the HDF-EOS output file from a PGE. See Section 12.1 for how to inspect the metadata. An HDF-EOS file is defined a file produced using the SDP Toolkit metadata tools and having, at a minimum, the ECS core metadata.

The Activity Checklist table that follows provides an overview of the process for viewing product data with the EOSView Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.2-1 Viewing Product Data with the EOSView Tool - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the EOSView GUI.	(I) 14.2	
2	SSI&T	Invoke the Open GUI.	(I) 14.2	
3	SSI&T	Select the directory containing the HDF-EOS file(s) to examine.	(I) 14.2	
4	SSI&T	Select the HDF-EOS file to examine.	(I) 14.2	
5	SSI&T	Select the HDF object in the file to examine.	(I) 14.2	
6	SSI&T	Display data for object.	(I) 14.2.1 (I) 14.2.2 (I) 14.2.3	
7	SSI&T	Repeat steps 5 and 6 for each object to examine.	(I) 14.2	
8	SSI&T	Close the HDF-EOS file.	(I) 14.2	
9	SSI&T	Quit the EOSView GUI.	(I) 14.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools.
2. EOSView has been properly installed and is accessible to the user.

To view product data with the EOSView tool, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Product Examination**, then **EOSView**.
- The EOSView GUI will be displayed.

2 In the GUI labeled **EOSView - EOSView Main Window**, click on the **File** menu and select **Open.**
- The **Filter** GUI will be displayed.

3 In the **Filter** subwindow, enter full path name and file name wildcard template. For example, enter ***/home/MyDirectory/MySubdirectory/****.
- The ***/home/MyDirectory/MySubdirectory/**** represents the location to the directory containing the HDF-EOS files to examine.
- The asterisk (*) is a wildcard template that represents all files in that directory; other wildcard templates can narrow the search further, *e.g.* ***.hdf**.
- Use the **Directories** field to further select the correct directory.
- Files found matching the wildcard template in the chosen directory will be displayed in **Files** subwindow.

4 In the **Files** subwindow, click on the file name of the HDF-EOS file to examine. Then click on the **OK** button.
- A GUI labeled **EOSView - *MyOutputFile.hdf*** will be displayed where *MyOutputFile.hdf* is the file name of the file chosen in step 3.
- Be patient - this GUI may take some time to appear, particularly for large files.
- Once displayed, a list of HDF objects will appear in the main window. If nothing is listed, it means that no HDF objects were found within the file.

5 In the GUI labeled **EOSView - *MyOutputFile.hdf***, double click on an object listed for which data is to be inspected.

6 Go to procedure depending upon the type of the object selected in step 5.
- Proceed to Section 14.2.1 if object is an HDF Image.
- Proceed to Section 14.2.2 if object is an HDF-EOS Grid.
- Proceed to Section 14.2.3 if object is an HDF-EOS Swath.
- Proceed to Section 14.2.4 if object is an HDF SDS.
- Proceed to Section 14.2.5 is object is in another format.

7 Repeat steps 5 and 6 for each HDF object within the selected HDF-EOS file for which data is to be examined.

8 In the GUI labeled **EOSView - *MyOutputFile.hdf***, click on the **File** menu and select **Close**.
- The **EOSView - *MyOutputFile.hdf*** GUI will disappear.
- Be patient - this GUI may take some time to disappear, particularly for large files.

9 In the GUI labeled **EOSView - EOSView Main Window**, click on the **File** menu and select **Exit**.

- The **EOSView - EOSView Main Window** GUI will disappear.

Table 14.2-2 Viewing Product Data with the EOSView Tool - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Product Examination → EOSView	(No action)
2	File → Open	(No action)
3	*/home/MyDirectory/MySubdirectory/**	select file filter
4	OK	select file
5	select object	double click
6	(No entry)	go to procedure 14.2.1, 14.2.2, 14.2.3, 14.2.4, or 14.2.5
7	(No entry)	repeat 5 and 6
8	File → Close	(No action)
9	File → Exit	(No action)

14.2.1 Viewing HDF Image Objects

This procedure describes how to use the EOSView tool to view science Images (typically, browse images) in the HDF-EOS output file from a PGE. See Section 14.2 for how to select an HDF Image object within an HDF-EOS file. An HDF-EOS file is defined a file produced using the SDP Toolkit metadata tools and having, at a minimum, the ECS core metadata.

Once an HDF Image is displayed, a number of options are available. These include colorization, zooming, panning, and animation. Each is described in the procedures below as an optional step.

EOSView will display and HDF Image in its referenced default color palette if the file contains one. If the display looks “blacked out”, it may mean that no default color palette is available or referenced by the Image object. In this case, a palette will have to be selected by the user.

Zooming allows both zoom in and zoom out according to a bilinear interpolation or nearest neighbor resampling method. Bilinear interpolation involves averaging to produce a “smoothed” display appropriate for grey scale band Images or derived geophysical parameter Images (*e.g.* temperature, vegetation index, albedo). Nearest neighbor produces a non-smoothed display appropriate for derived thematic Image maps (*e.g.* land cover, masks).

Panning allows user to select what portion of a zoomed Image is being displayed.

Animation allows multiple Images to be displayed in an animated fashion in a number of modes. Stop-at-End mode allows Images to be displayed to the end just once. Continuous Run mode lets the animation run through the Images repeatedly and Bounce mode does a forward/reverse run through any set of Images repeatedly. All animation runs can be stopped at any time and then resumed in either the forward or reverse directions. The speed of an animation can also be adjusted as well.

The Activity Checklist table that follows provides an overview of the process for viewing HDF Image objects using the EOSView Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.2.1-1 Viewing HDF Image Objects - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Select the Image object to be viewed.	(I) 14.2.1	
2	SSI&T	Optionally, select the color palette of the Image.	(I) 14.2.1	
3	SSI&T	Optionally, zoom the Image.	(I) 14.2.1	
4	SSI&T	Optionally, pan the zoomed Image.	(I) 14.2.1	
5	SSI&T	End the colorization, zooming, or panning session with the Image.	(I) 14.2.1	
6	SSI&T	Optionally, animate multiple Images.	(I) 14.2.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools and that at least one object is an HDF image (RIS8, RIS24, *i.e.* Browse data).
2. EOSView has been properly installed and is accessible to the user.
3. The HDF-EOS file has been read into EOSView by the procedures described in Section 14.2.

To view an HDF-EOS Image object with the EOSView tool, execute the procedure steps that follow:

1 In the GUI labeled **EOSView - *MyOutputFile.hdf***, double click on an Image object listed for which data is to be inspected.

- A GUI labeled **EOSView - Image Display Window - *MyImageObject*** will be displayed where *MyImageObject* will be replaced by the name of the object selected as listed. For example, **EOSView - Image Display Window - Image [512x512] ref=2 (palette)**.
- Be patient - this GUI may take some time to appear, particularly for large files.

2 Optional colorization. In the GUI labeled **EOSView - Image Display Window - *MyImageObject***, click on the **Palette** menu, then select **Select** and then select one of the palettes listed: **Default**, **Greyscale**, **Antarctica**, **Rainbow**, or **World Colors**.

- The selection of palette will not result in any change to the data in the file or to the object's default palette.
- This selection may be repeated until the desired palette is chosen.

3 Optional zooming. In the GUI labeled **EOSView - Image Display Window - *MyImageObject***, click on the **Zooming** menu, then select **Select** and then select one of the resampling methods listed: **Bilinear Interpolation** or **Nearest Neighbor**. Then click on the **Zoom In** or **Zoom Out** buttons to apply the method.

- The selection of resampling method and zoom will not result in any change to the data in the file.
- The zooming options may be repeated as desired.

4 Optional panning while zooming. In the GUI labeled **EOSView - Image Display Window - *MyImageObject***, a thumbnail representation of the entire Image will be displayed in the subwindow labeled **Pan Window**. A hollow rectangle on the thumbnail indicates that portion of the Image that is being displayed in the main window. Use the mouse left button to click and drag the rectangle to a new location on the thumbnail image.

- The portion of the zoomed Image shown in the main window will be the portion indicated by the hollow rectangle on the thumbnail image.
- The panning will not result in any change to the data in the file.
- The panning option may be repeated as desired.

5 To end the session with colorization, zooming, or panning, in the GUI labeled **EOSView - Image Display Window - *MyImageObject***, click on the **File** menu and select **Close**.

- The **EOSView - Image Display Window - *MyImageObject*** GUI will disappear.

6 Optional animation. In the GUI labeled **EOSView - *MyOutputFile.hdf***, click on the **Options** menu, then select **Animated images**.

- A GUI labeled **EOSView - Image Animation Window - *MyOutputFile.hdf*** will be displayed.
- Be patient - this GUI may take some time to appear, particularly for large files.
- Optionally, click on the **Palette** menu to select a palette.
- Optionally, click on the **Options** menu and then select **Mode** to select how the animation is to be run. Choose **Stop at end**, **Continuous run**, or **Bounce**.
- Click on the Stop button, denoted by the **||** symbol (center) to halt the animation.
- Resume the animation by clicking on either the Forward Play (denoted by the **>>** symbol) or the Reverse Play (denoted by the **<<** symbol).
- There is also Forward Increment (**>|**) and Reverse Increment (**|<**) button.
- The animation speed may be adjusted by moving the **Speed** slider in either the "+" or "-" direction.
- To end animation session, click on the **File** menu and then select **Close**.

Table 14.2.1-2 Viewing HDF Image Objects - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	select Image object	double click
2	(Optional) Palette → Select → Default | Greyscale | Antarctica | Rainbow | World Colors	choose palette
3	(Optional) Zooming → Select → Bilinear Interpolation | Nearest Neighbor	choose resampling mode
4	(Optional) Pan Window	click and drag
5	File → Close	(No action)
6	(Optional) Options → Animated images	adjust as desired

14.2.2 Viewing HDF-EOS Grid Objects

This procedure describes how to use the EOSView tool to view science data in the HDF-EOS output file that are in HDF-EOS Grid format. These are generally the science data and not browse images). See Section 14.2 for how to select an HDF-EOS Grid object within an HDF-EOS file. An HDF-EOS file is defined a file produced using the SDP Toolkit metadata tools and having, at a minimum, the ECS core metadata.

The Activity Checklist table that follows provides an overview of the process for viewing HDF-EOS Grid objects using the EOSView Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.2.2-1 Viewing HDF-EOS Grid Objects - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Select the Grid object to be viewed.	(I) 14.2.2	
2	SSI&T	Select the data field to display.	(I) 14.2.2	
3	SSI&T	Display table of data values.	(I) 14.2.2	
4	SSI&T	Make image from the data values.	(I) 14.2.2	
5	SSI&T	Optionally, select the color palette of the Grid data.	(I) 14.2.2	
6	SSI&T	Optionally, zoom the Grid data.	(I) 14.2.2	
7	SSI&T	Optionally, pan the zoomed Grid data.	(I) 14.2.2	
8	SSI&T	End session with displayed Grid data.	(I) 14.2.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools and that at least one object is an HDF-EOS Grid.
2. EOSView has been properly installed and is accessible to the user.
3. The HDF-EOS file has been read into EOSView by the procedures described in Section 14.2.

To view an HDF-EOS Grid object with the EOSView tool, execute the procedure steps that follow:

1 In the GUI labeled **EOSView -** ***MyOutputFile.hdf***, double click on an Grid object listed for which data is to be inspected.

- A GUI labeled **EOSView - Grid Select** will be displayed.
- Be patient - this GUI may take some time to appear, particularly for large Grid objects.
- Clicking on the **Grid Information** checkbox and then clicking on the **OK** button displays grid information about the selected Grid object such as X-Dimension value, Y-Dimension value, Upper Left Point values, and Lower Right Point values.
- Clicking on the **Projection Information** checkbox and then clicking on the **OK** button displays projection information about the selected Grid object such as the Projection Name and the Zone Code.
- Clicking on the **Dimensions** checkbox and then clicking on the **OK** button displays dimension information about the selected Grid object such as the Dimension Names and Sizes.
- Clicking on the **Attributes** checkbox and then clicking on the **OK** button displays attribute information about the selected Grid object such as the attribute names and values.

2 In the GUI labeled **EOSView - Grid Select**, click on the **Data Fields** checkbox and then click on the **OK** button. Then double click on one of the data fields listed.

- A GUI labeled **EOSView - Grid -** ***GridObjectName*** **- Start/Stride/Edge** will be displayed where *GridObjectName* will be replaced by the name of the Grid object selected in step 1.

3 In the GUI labeled **EOSView - Grid -** ***GridObjectName*** **- Start/Stride/Edge**, click on the checkboxes for both **YDim** and **XDim** and then click on the **OK** button.

- A GUI labeled ***MyDataField*** will be displayed where *MyDataField* will be replaced by the name of the data field selected in step 2.
- The data will be displayed in this GUI in the form of a table of values.

4 In the GUI labeled ***MyDataField***, click on the **File** menu and then select **Make Image**. Optionally adjust the default minimum and maximum data values and then click on the **Continue** button.

- Adjusting the minimum and maximum data values will affect only the display (contrast) and not the actual data in the file.
- A GUI labeled **EOSView - Swath/Grid Image** will appear, displaying the data field in image form.

5 Optional colorization. In the GUI labeled **EOSView - Swath/Grid Image**, click on the **Palette** menu, then select **Select** and then select one of the palettes listed: **Default**, **Greyscale**, **Antarctica**, **Rainbow**, or **World Colors**.

- The selection of palette will not result in any change to the data in the file or to the object's default palette.
- This selection may be repeated until the desired palette is chosen.

6 Optional zooming. In the GUI labeled **EOSView - Swath/Grid Image**, click on the **Zooming** menu, then select **Select** and then select one of the resampling methods listed: **Bilinear Interpolation** or **Nearest Neighbor**. Then click on the **Zoom In** or **Zoom Out** buttons to apply the method.

- The selection of resampling method and zoom will not result in any change to the data in the file.
- The zooming options may be repeated as desired.

7 Optional panning while zooming. In the GUI labeled **EOSView - Swath/Grid Image**, a thumbnail representation of the entire Image will be displayed in the subwindow labeled **Pan Window**. A hollow rectangle on the thumbnail indicates that portion of the Image that is being displayed in the main window. Use the mouse left button to click and drag the rectangle to a new location on the thumbnail image.

- The portion of the zoomed Image shown in the main window will be the portion indicated by the hollow rectangle on the thumbnail image.
- The panning will not result in any change to the data in the file.
- The panning option may be repeated as desired.

8 To end the session with displaying Grid object, in the GUI labeled **EOSView - Swath/Grid**, click on the **File** menu and select **Close**.

- The **EOSView - Swath/Grid** GUI will disappear.

Table 14.2.2-2 Viewing HDF-EOS Grid Objects - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	select Grid object	double click
2	Data Fields → OK	(No action)
3	Ydim, Xdim → OK	(No action)
4	File → Make image → Continue	(No action)

Table 14.2.2-2 Viewing HDF-EOS Grid Objects - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
5	(Optional) Palette → Select → Default | Greyscale | Antarctica | Rainbow | World Colors	choose palette
6	(Optional) Zooming → Select → Bilinear Interpolation | Nearest Neighbor	choose resampling mode
7	(Optional) Pan Window	click and drag
8	File → Close	(No action)

14.2.3 Viewing HDF-EOS Swath Objects

This procedure describes how to use the EOSView tool to view science data in the HDF-EOS output file that are in HDF-EOS Swath format. These are generally the science data and not browse images). See Section 14.2 for how to select an HDF-EOS Swath object within an HDF-EOS file. An HDF-EOS file is defined a file produced using the SDP Toolkit metadata tools and having, at a minimum, the ECS core metadata.

The Activity Checklist table that follows provides an overview of the process for viewing HDF-EOS Swath objects using the EOSView Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.2.3-1 Viewing HDF-EOS Swath Objects - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Select the Swath object to be viewed.	(I) 14.2.3	
2	SSI&T	Select the data field to display.	(I) 14.2.3	
3	SSI&T	Display table of data values.	(I) 14.2.3	
4	SSI&T	Make image from the data values.	(I) 14.2.3	
5	SSI&T	Optionally, select the color palette of the Swath data.	(I) 14.2.3	
6	SSI&T	Optionally, zoom the Swath data.	(I) 14.2.3	
7	SSI&T	Optionally, pan the zoomed Swath data.	(I) 14.2.3	
8	SSI&T	End session with displayed Swath data.	(I) 14.2.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools and that at least one object is an HDF-EOS Swath.
2. EOSView has been properly installed and is accessible to the user.
3. The HDF-EOS file has been read into EOSView by the procedures described in Section 14.2.

To view an HDF-EOS Swath object with the EOSView tool, execute the procedure steps that follow:

1 In the GUI labeled **EOSView - *MyOutputFile.hdf***, double click on a Swath object listed for which data is to be inspected.

- A GUI labeled **EOSView - Swath Select** will be displayed.
- Be patient - this GUI may take some time to appear, particularly for large Swath objects.
- Clicking on the **Dimensions** checkbox and then clicking on the **OK** button displays dimension information about the selected Swath object such as Dimension Names and Sizes.
- Clicking on the **Geolocation Mappings** checkbox and then clicking on the **OK** button displays geolocation information about the selected Swath object such as the Geolocation Dimensions, Data Dimensions, Offsets, and Increments.
- Clicking on the **Indexed Mappings** checkbox and then clicking on the **OK** button displays index mapping information about the selected Swath.
- Clicking on the **Geolocation Fields** checkbox and then clicking on the **OK** button displays geolocation fields information about the selected Swath object.
- Clicking on the **Attributes** checkbox and then clicking on the **OK** button displays attribute information about the selected Swath object.

2 In the GUI labeled **EOSView - Swath Select**, click on the **Data Fields** checkbox and then click on the **OK** button. Then double click on one of the data fields listed.

- A GUI labeled **EOSView - Swath - *SwathObjectName* - Start/Stride/Edge** will be displayed where *SwathObjectName* will be replaced by the name of the Swath object selected in step 1.

3 In the GUI labeled **EOSView - Swath - *SwathObjectName* - Start/Stride/Edge**, click on the checkboxes for both **ScanLineTra** and **PixelsXtrac** and then click on the **OK** button.

- A GUI labeled ***MyDataField*** will be displayed where *MyDataField* will be replaced by the name of the data field selected in step 2.
- The data will be displayed in this GUI in the form of a table of values.

4 In the GUI labeled ***MyDataField***, click on the **File** menu and then select **Make Image**. Optionally adjust the default minimum and maximum data values and then click on the **Continue** button.

- Adjusting the minimum and maximum data values will affect only the display (contrast) and not the actual data in the file.
- A GUI labeled **EOSView - Swath/Grid Image** will appear, displaying the data field in image form.

5 Optional colorization. In the GUI labeled **EOSView - Swath/Grid Image**, click on the **Palette** menu, then select **Select** and then select one of the palettes listed: **Default**, **Greyscale**, **Antarctica**, **Rainbow**, or **World Colors**.

- The selection of palette will not result in any change to the data in the file or to the object's default palette.
- This selection may be repeated until the desired palette is chosen.

6 Optional zooming. In the GUI labeled **EOSView - Swath/Grid Image**, click on the **Zooming** menu, then select **Select** and then select one of the resampling methods listed: **Bilinear Interpolation** or **Nearest Neighbor**. Then click on the **Zoom In** or **Zoom Out** buttons to apply the method.

- The selection of resampling method and zoom will not result in any change to the data in the file.
- The zooming options may be repeated as desired.

7 Optional panning while zooming. In the GUI labeled **EOSView - Swath/Grid Image**, a thumbnail representation of the entire Image will be displayed in the subwindow labeled **Pan Window**. A hollow rectangle on the thumbnail indicates that portion of the Image that is being displayed in the main window. Use the mouse left button to click and drag the rectangle to a new location on the thumbnail image.

- The portion of the zoomed Image shown in the main window will be the portion indicated by the hollow rectangle on the thumbnail image.
- The panning will not result in any change to the data in the file.
- The panning option may be repeated as desired.

8 To end the session with displaying Swath object, in the GUI labeled **EOSView - Swath/Grid**, click on the **File** menu and select **Close**.

- The **EOSView - Swath/Grid** GUI will disappear.

Table 14.2.3-2 Viewing HDF-EOS Swath Objects - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	select Swath object	double click
2	Data Fields → OK	(No action)
3	ScanLineTra, PixelsXtrac → OK	(No action)
4	File → Make image → Continue	(No action)

Table 14.2.3-2 Viewing HDF-EOS Swath Objects - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
5	(Optional) Palette → Select → Default | Greyscale | Antarctica | Rainbow | World Colors	choose palette
6	(Optional) Zooming → Select → Bilinear Interpolation | Nearest Neighbor	choose resampling mode
7	(Optional) Pan Window	click and drag
8	File → Close	(No action)

14.2.4 Viewing HDF SDS Objects

This procedure describes how to use the EOSView tool to view science data in the HDF-EOS output file that are in HDF SDS (standard HDF science data set) format. These are generally the science data and not browse images). See Section 14.2 for how to select an HDF SDS object within an HDF-EOS file. An HDF-EOS file is defined a file produced using the SDP Toolkit metadata tools and having, at a minimum, the ECS core metadata.

The Activity Checklist table that follows provides an overview of the process for viewing HDF SDS objects using the EOSView Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.2.4-1 Viewing HDF SDS Objects - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Select the SDS object to be viewed.	(I) 14.2.4	
2	SSI&T	Select two dimensions and then display table.	(I) 14.2.4	
3	SSI&T	Make image from the data values.	(I) 14.2.4	
4	SSI&T	Optionally, select the color palette of the SDS data.	(I) 14.2.4	
5	SSI&T	Optionally, zoom the SDS data.	(I) 14.2.4	
6	SSI&T	Optionally, pan the zoomed SDS data.	(I) 14.2.4	
7	SSI&T	End session with displayed SDS data.	(I) 14.2.4	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools and that at least one object is an HDF SDS.
2. EOSView has been properly installed and is accessible to the user.
3. The HDF-EOS file has been read into EOSView by the procedures described in Section 14.2.

To view an HDF SDS object with the EOSView tool, execute the procedure steps that follow:

1 In the GUI labeled **EOSView - *MyOutputFile.hdf***, double click on a SDS object listed for which data is to be inspected.

- A GUI labeled **EOSView - Multi-Dimension SDS** will be displayed.
- A number of checkboxes will be displayed, one for each of the dimensions in the selected SDS (there will be at least two, an X and a Y).
- Be patient - this GUI may take some time to appear, particularly for large SDS objects.

2 In the GUI labeled **EOSView - Multi-Dimension SDS**, click on two of the dimension checkboxes and then click on the **Table** button. Then double click on one of the data fields listed.

- A GUI labeled ***MySDS*** will be displayed where *MySDS* will be replaced by the name of the SDS object selected in step 1.

3 In the GUI labeled ***MySDS***, click on the **File** menu and then select **Make Image**. Optionally adjust the default minimum and maximum data values and then click on the **Continue** button.

- Adjusting the minimum and maximum data values will affect only the display (contrast) and not the actual data in the file.
- A GUI labeled **EOSView - Image Display Window - *MySDS*** will appear, displaying the data field in image form.

4 Optional colorization. In the GUI labeled **EOSView - Image Display Window - *MySDS***, click on the **Palette** menu, then select **Select** and then select one of the palettes listed: **Default**, **Greyscale**, **Antarctica**, **Rainbow**, or **World Colors**.

- The selection of palette will not result in any change to the data in the file or to the object's default palette.
- This selection may be repeated until the desired palette is chosen.

5 Optional zooming. In the GUI labeled **EOSView - Image Display Window - *MySDS***, click on the **Zooming** menu, then select **Select** and then select one of the resampling methods listed: **Bilinear Interpolation** or **Nearest Neighbor**. Then click on the **Zoom In** or **Zoom Out** buttons to apply the method.

- The selection of resampling method and zoom will not result in any change to the data in the file.
- The zooming options may be repeated as desired.

6 Optional panning while zooming. In the GUI labeled **EOSView - Image Display Window - *MySDS***, a thumbnail representation of the entire Image will be displayed in the subwindow labeled **Pan Window**. A hollow rectangle on the thumbnail indicates that portion of the Image that is being displayed in the main window. Use the mouse left button to click and drag the rectangle to a new location on the thumbnail image.

- The portion of the zoomed Image shown in the main window will be the portion indicated by the hollow rectangle on the thumbnail image.
- The panning will not result in any change to the data in the file.
- The panning option may be repeated as desired.

7 To end the session with displaying Swath object, in the GUI labeled **EOSView - Image Display Window - *MySDS***, click on the **File** menu and select **Close**.

- The **EOSView - Image Display Window - *MySDS*** GUI will disappear.

Table 14.2.4-2 Viewing HDF SDS Objects - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	select SDS object	double click
2	dimensions → Table	(No action)
3	File → Make image → Continue	(No action)
4	(Optional) Palette → Select → Default | Greyscale | Antarctica | Rainbow | World Colors	choose palette
5	(Optional) Zooming → Select → Bilinear Interpolation | Nearest Neighbor	choose resampling mode
6	(Optional) Pan Window	click and drag
7	File → Close	(No action)

14.3 Viewing Product Data with the IDL Tool

This procedure describes how to use the IDL (Interactive Data Language) COTS tool to inspect the data in the output file from a PGE. These procedures are geared toward binary and ASCII formats, but can be extended to other formats supported by IDL including HDF, NetCDF, and PGE. Consult the IDL references for details on these other formats. See Section 14.2 for how to inspect the science data and metadata in an HDF-EOS file.

The major activities addresses here include creating an image display (Section 14.3.1), saving an image display (Section 14.3.2), creating a plot display (Section 14.3.3), and saving a plot display (Section 14.3.4).

The Activity Checklist table that follows provides an overview of the process for viewing product data with the IDL Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.3-1 Viewing Product Data with the IDL Tool - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke the IDL tool.	(I) 14.3	
2	SSI&T	Select an IDL activity.	(I) 14.3.1 (I) 14.3.2 (I) 14.3.3 (I) 14.3.4 (I) 14.3.5	
3	SSI&T	Quit the IDL session.	(I) 14.3	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The output file is binary, ASCII, or one of the other IDL supported data formats.
2. IDL has been properly installed and is accessible to the user.

To view product data with the IDL tool, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Product Examination**, then **IDL**.
- An xterm (on the AIT Sun) will be displayed within which the IDL command interpreter will be run.

2 Go to procedure depending upon the activity to perform.
- Proceed to Section 14.3.1 to create an image display.
- Proceed to Section 14.3.2 to save an image display.
- Proceed to Section 14.3.3 to create a plot display.
- Proceed to Section 14.3.4 to save a plot display.

3 To end the IDL session, close any display windows remaining, then at the IDL prompt type **quit**, press **Return**.
- The IDL session will be closed.

Table 14.3-2 Viewing Product Data with the IDL Tool - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Product Examination → IDL	(No action)
2	(No entry)	choose an IDL activity
3	quit	press Return

14.3.1 Creating an Image Display Using IDL

This procedure describes how to use the IDL Tool to create an image display. The next procedure, Section 14.3.2, describes how to save an image display (once created) to either a data file or a graphic file. For creating and saving plot displays, see Sections 14.3.3 and 14.3.4, respectively.

The Activity Checklist table that follows provides an overview of the process for creating an image display using the IDL Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.3.1-1 Creating an Image Display Using IDL - Activity Checklist

Order	Role	Task	Section	Complete?
Binary Data:				
1	SSI&T	Open binary data input file.	(I) 14.3.1	
2	SSI&T	Create an image array for the data.	(I) 14.3.1	
3	SSI&T	Read binary data into image array.	(I) 14.3.1	
4	SSI&T	Display the image.	(I) 14.3.1	
5	SSI&T	Load a color table.	(I) 14.3.1	
6	SSI&T	Close the input file.	(I) 14.3.1	
ASCII Data:				
1	SSI&T	Open ASCII data input file.	(I) 14.3.1	
2	SSI&T	Create an image array for the data.	(I) 14.3.1	
3	SSI&T	Read ASCII data into image array.	(I) 14.3.1	
4	SSI&T	Display the image.	(I) 14.3.1	
5	SSI&T	Load a color table.	(I) 14.3.1	
6	SSI&T	Close the input file.	(I) 14.3.1	
PGM Data:				
1	SSI&T	Read PGM formatted file into image array.	(I) 14.3.1	
2	SSI&T	Load a color table.	(I) 14.3.1	
3	SSI&T	Display the image.	(I) 14.3.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. IDL is running (see Section 14.3).
2. The PGE output file to be examined is of an IDL-supported type/format (if in doubt, consult the *IDL Reference Guide*).
3. For binary files, data is assumed to be 8-bit characters. Follow the steps listed under **Binary Data**.
4. For ASCII files, data is assumed to be comma-delimited characters. Follow the steps listed under **ASCII Data**.
5. For PGM (Portable Grey Map) formatted files, dimension information is carried in the file header and therefore it does not need to be known. Follow the steps listed under **PGM Data**.

To create an image display using IDL, execute the procedure steps that follow:

Binary Data:

1 At the IDL prompt, type **OPENR,1,(‘*MyBinaryFilename*’)**, press **Return**.
- The ***MyBinaryFilename*** is the full path name and file name of the binary data file of known dimensions to read in.
- The single quotes (‘) must be included around the path/file name.
- The **1** is the logical unit number.

2 At the IDL prompt, type ***MyImage*=BYTARR(*dim1,dim2*)**, press **Return**.
- The ***MyImage*** is the name to be given to the image once created.
- The ***dim1*** and ***dim2*** are the dimensions of the input data.

3 At the IDL prompt, type **READU,1,*MyImage***, press **Return**.

4 At the IDL prompt, type **TV,*MyImage***, press **Return**.
- The image, ***MyImage***, should then be displayed.

5 At the IDL prompt, type **LOADCT,3**, press **Return**.
- This command loads color table number 3. Other color tables are available; refer to the *IDL Reference Guide* for more details.

6 At the IDL prompt, type **CLOSE,1**, press **Return**.
- This closes logical unit 1.
- Always close logical units or an error will result the next time an access is attempted.

ASCII Data:

1 At the IDL prompt, type **OPENR,1,('*MyASCIIfilename*')**, press **Return**.

- The ***MyASCIIfilename*** is the full path name and file name of the ASCII data file of known dimensions to read in.
- The single quotes (') must be included around the path/file name.
- The **1** is the logical unit number.

2 At the IDL prompt, type ***MyImage*=BYTARR(*dim1*,*dim2*)**, press **Return**.

- The ***MyImage*** is the name to be given to the image once created.
- The ***dim1*** and ***dim2*** are the dimensions of the input data.

3 At the IDL prompt, type **READF,1,*MyImage***, press **Return**

4 At the IDL prompt, type **TV,*MyImage***, press **Return**.

- The image, ***MyImage***, should then be displayed.

5 At the IDL prompt, type **LOADCT,3**, press **Return**.

- This command loads color table number 3. Other color tables are available; refer to the *IDL Reference Guide* for more details.

6 At the IDL prompt, type **CLOSE,1**, press **Return**.

- This closes logical unit 1.
- Always close logical units or an error will result the next time an access is attempted.

PGM Data:

1 At the IDL prompt, type **READ_PPM,"*MyPGMfilename*",*MyImage*,r,g,b**, press **Return**.

- The ***MyPGMfilename*** is the full path name and file name of the PGM formatted data file.
- The double quotes (") must be included around the path/file name.
- The ***MyImage*** is the name to be given to the image created.

2 At the IDL prompt, type **TVLCT,r,g,b**, press **Return**.

- Note that r,g,b color table syntax is used for most formatted file types in IDL.

3 At the IDL prompt, type **TV,*MyImage***, press **Return**.

- The image, ***MyImage***, should then be displayed.

Table 14.3.1-2 Creating an Image Display Using IDL - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
Binary Data:		
1	OPENR,1,('*MyBinaryFilename*')	press Return
2	*MyImage*=BYTARR(*dim1*,*dim2*)	press Return
3	READU,1,*MyImage*	press Return
4	TV,*MyImage*	press Return
5	LOADCT,3	press Return
6	CLOSE,1	press Return
ASCII Data:		
1	OPENR,1,('*MyASCIIfilename*')	press Return
2	*MyImage*=BYTARR(*dim1*,*dim2*)	press Return
3	READF,1,*MyImage*	press Return
4	TV,*MyImage*	press Return
5	LOADCT,3	press Return
6	CLOSE,1	press Return
PGM Data:		
1	READ_PPM,"*MyPGMfilename*",*MyImage*,r,g,b	press Return
2	TVLCT,r,g,b	press Return
3	TV,*MyImage*	press Return

14.3.2 Saving an Image Display Using IDL

This procedure describes how to use the IDL Tool to save an image display. The previous procedure, Section 14.3.1, describes how to create an image display. For creating and saving plot displays, see Sections 14.3.3 and 14.3.4, respectively.

The Activity Checklist table that follows provides an overview of the process for saving an image display using the IDL Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.3.2-1 Saving an Image Display Using IDL - Activity Checklist

Order	Role	Task	Section	Complete?
Binary Data:				
1	SSI&T	Open binary data output file.	(I) 14.3.2	
2	SSI&T	Write the output file.	(I) 14.3.2	
3	SSI&T	Close the output file.	(I) 14.3.2	
ASCII Data:				
1	SSI&T	Open ASCII data output file.	(I) 14.3.2	
2	SSI&T	Write the output file.	(I) 14.3.2	
3	SSI&T	Close the output file.	(I) 14.3.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. IDL is running (see Section 14.3).
2. The image display is to be saved in a binary (8-bit) or ASCII (comma-delimited characters) format. For other output formats, consult the *IDL Reference Guide*.
3. For binary files, data will be output as 8-bit characters. Follow the steps listed under **Binary Data**.
4. For ASCII files, data will be output as comma-delimited characters. Follow the steps listed under **ASCII Data**.

To save an image display using IDL, execute the procedure steps that follow:

Binary Data:

1 At the IDL prompt, type **OPENW,1,('*MyBinaryFilename.bin*')**, press **Return**.

- The ***MyBinaryFilename.bin*** is the full path name and file name of the binary data file to write out.
- The single quotes (') must be included around the path/file name.
- The **1** is the logical unit number.

2 At the IDL prompt, type **WRITEU,1,*MyImage***, press **Return**.

- The ***MyImage*** is the name of the image to save.

3 At the IDL prompt, type **CLOSE,1**, press **Return**.

- This closes logical unit 1.
- Always close logical units or an error will result the next time an access is attempted.

ASCII Data:

1 At the IDL prompt, type **OPENW,1,('*MyASCIIfilename.asc*')**, press **Return**.
- The ***MyASCIIfilename.asc*** is the full path name and file name of the binary data file to write out.
- The single quotes (') must be included around the path/file name.
- The **1** is the logical unit number.

2 At the IDL prompt, type **PRINTF,1,*MyImage***, press **Return**.
- The ***MyImage*** is the name of the image to save.

3 At the IDL prompt, type **CLOSE,1**, press **Return**.
- This closes logical unit 1.
- Always close logical units or an error will result the next time an access is attempted.

Table 14.3.2-2 Saving an Image Display Using IDL - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
Binary Data:		
1	OPENW,1,('*MyBinaryFilename.bin*')	press Return
2	WRITEU,1,*MyImage*	press Return
3	CLOSE,1	press Return
ASCII Data:		
1	OPENW,1,('*MyASCIIfilename.asc*')	press Return
2	PRINTF,1,*MyImage*	press Return
3	CLOSE,1	press Return

14.3.3 Creating a Plot Display Using IDL

14.3.4 Saving a Plot Display Using IDL

14.4 Raster Mapping Fundamentals

This procedure describes how to use the IDL Tool to perform basic raster mapping functions. These are spatial functions involving map projections, but do not include surface modeling (also called "2.5D") or two-dimensional spectral functions.

The Activity Checklist table that follows provides an overview of the process for raster mapping fundamentals using the IDL Tool. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 14.4-1 Raster Mapping Fundamentals - Activity Checklist

Order	Role	Task	Section	Complete?
Global Data Set Image:				
1	SSI&T	Display the global data set image.	(I) 14.4	
2	SSI&T	Set the map projection.	(I) 14.4	
3	SSI&T	Create a new image using map projection.	(I) 14.4	
4	SSI&T	Display new image.	(I) 14.4	
5	SSI&T	Optionally, overlay Lat/Lon grid.	(I) 14.4	
6	SSI&T	Optionally, overlay world coastlines.	(I) 14.4	
Sub-Global Data Set Image:				
1	SSI&T	Display the sub-global data set image.	(I) 14.4	
2	SSI&T	Set the map projection with limits.	(I) 14.4	
3	SSI&T	Create a new image within limits using map projection.	(I) 14.4	
4	SSI&T	Display new image.	(I) 14.4	
5	SSI&T	Optionally, overlay Lat/Lon grid.	(I) 14.4	
6	SSI&T	Optionally, overlay world coastlines.	(I) 14.4	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. IDL is running (see Section 14.3).
2. For a global data set image, one having geocentric-LLR coordinates defined from -180 to 180 degrees East Longitude and 90 to -90 degrees North Latitude, follow the steps listed under **Global Data Set Image**.
3. For a sub-global data set image, one having geocentric-LLR coordinates defined for subintervals of longitude and latitude (*e.g.* from -88 to -77 degrees East Longitude and 23 to 32 degrees North Latitude), follow the steps listed under **Sub-Global Data Set Image.**

To perform basic raster mapping for the cases listed above using IDL, execute the procedure steps that follow:

Global Data Set Image:

1 At the IDL prompt, type **TV,*MyImage***, press **Return**.

- The ***MyImage*** is the image name of the global image data set.
- The image, ***MyImage***, should then be displayed.

2 At the IDL prompt, type **MAP_SET,/ORTHOGRAPHIC**, press **Return**.

- IDL also supports other map projections. Refer to *IDL Reference Guide*.

3 At the IDL prompt, type ***MyNewImage*=MAP_IMAGE(*MyImage*,startx,starty,/*BILIN*)**, press **Return**.

- The ***MyNewImage*** is the name to assign to the resulting image.
- The ***MyImage*** is the name of the original global image data set.

4 At the IDL prompt, type **TV,*MyNewImage*,startx,starty**, press **Return**.

- The image ***MyNewImage*** should then be displayed.

5 Optional overlay Lat/Lon. At the IDL prompt, type **MAP_GRID**, press **Return**.

- This overlays Lat/Lon graticule onto ***MyNewImage***.

6 Optional overlay world coastlines. At the IDL prompt, type **MAP_CONTINENTS**, press **Return**.

- This overlays world coastlines onto ***MyNewImage***.

Sub-Global Data Set Image:

1 At the IDL prompt, type **TV,*MyImage***, press **Return**.

- The ***MyImage*** is the image name of the sub-global image data set.
- The image, ***MyImage***, should then be displayed.

2 At the IDL prompt, type **MAP_SET,/MERCATOR,LIMIT=[*lat1*,*lon1*,*lat2*,*lon2*]**, press **Return**.

- The ***lat1***, ***lon1***, ***lat2***, and ***lon2*** specify the latitude and longitude intervals of the sub-global image data set. For example, type **MAP_SET,/MERCATOR,LIMIT=[23,-88,32,-77]**, press **Return**.

3 At the IDL prompt, type ***MyNewImage*=MAP_IMAGE(*MyImage*,startx,starty,/*BILIN*.LATMIN=*lat1*,LATMAX=*lat2*,LONMIN=*lon1*,LONMAX=*lon2*)**, press **Return**.

- The ***MyNewImage*** is the name to assign to the resulting image.
- The ***MyImage*** is the name of the original global image data set.
- The ***lat1***, ***lon1***, ***lat2***, and ***lon2*** specify the latitude and longitude intervals of the sub-global image data set.

4 At the IDL prompt, type **TV,*MyNewImage*,startx,starty**, press **Return**.

- The image ***MyNewImage*** should then be displayed.

5 Optional overlay Lat/Lon. At the IDL prompt, type **MAP_GRID**, press **Return**.

- This overlays Lat/Lon graticule onto ***MyNewImage***.

6 Optional overlay world coastlines. At the IDL prompt, type **MAP_CONTINENTS**, press **Return**.

- This overlays world coastlines onto ***MyNewImage***.

Table 14.4-2 Raster Mapping Fundamentals - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
Global Data Set Image:		
1	TV,*MyImage*	press Return
2	MAP_SET,/ORTHOGRAPHIC	press Return
3	*MyNewImage*=MAP_IMAGE(*MyImage*,startx,starty,/*BILIN*)	press Return
4	TV,*MyNewImage*,startx,starty	press Return
5	(Optional) MAP_GRID	press Return
6	(Optional) MAP_CONTINENTS	press Return
Sub-Global Data Set Image:		
1	TV,*MyImage*	press Return
2	MAP_SET,/MERCATOR,LIMIT=[*lat1*,*lon1*,*lat2*,*lon2*]	press Return
3	*MyNewImage*=MAP_IMAGE(*MyImage*,startx,starty,/*BILIN*. LATMIN=*lat1*,LATMAX=*lat2*,LONMIN=*lon1*,LONMAX=*lon2*)	press Return
4	TV,*MyNewImage*,startx,starty	press Return
5	(Optional) MAP_GRID	press Return
6	(Optional) MAP_CONTINENTS	press Return

This page intentionally left blank.

15. Post-SSI&T Activities

15.1 Validating Inventory Metadata Updates

15.2 Reporting Problems

This section describes procedures for reporting problems that occur during SSI&T. Both science software and ECS problem reporting are handled with the same COTS tool, PureDDTS (Distributed Defect Tracking System) by PureAtria Software, Inc. At some DAACs, two separate copies of DDTS may be used with one copy configured for reporting ECS problems and the other configured for reporting science software problems. At some DAACs, the same copy of DDTS may be used for both tasks with science software problem reporting set up as a distinct "project" having a unique DDTS configuration. In any case, there will likely be DAAC unique features associated with problem reporting of science software. What follows is therefore a generic description.

The environment set up for DDTS is described in Section 15.2.1. Section 15.2.2 describes how to submit a new problem report. Section 15.2.3 describes how to modify an existing problem report.

15.2.1 Setting Up the Environment for DDTS

This procedure describes how to set up the environment for using DDTS. See Section 15.2.2 for how to use DDTS for submitting science software problem reports.

The Activity Checklist table that follows provides an overview of the process for setting up the environment for DDTS. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 15.2.1-1 Setting Up the Environment for DDTS - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke text editor with the .cshrc file.	(I) 15.2.1	
2	SSI&T	Add lines to the .cshrc file.	(I) 15.2.1	
3	SSI&T	Save the file and quit the editor	(I) 15.2.1	
4	SSI&T	Reinitialize the environment.	(I) 15.2.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The user has authorization to use the science software configured DDTS.
2. The C shell or a derivative (*e.g.* T shell) is the current user shell.

To set up the environment for using DDTS for science software reporting, execute the procedure steps that follow:

1 At a UNIX prompt on an AIT Sun, type **vi $HOME/.cshrc**, press **Return.**

- This command invokes the *vi* editor and reads in the .cshrc file from the user's home directory.
- Any text editor may be used such as *emacs*. For example, **emacs $HOME/.cshrc**, press **Return**.

2 In the editor add the following lines if not already there:

```
#DDTS NCR Tool path
if (“ `echo $path | grep /home/ddts/bin`” == “ “) then
   set path=($path /home/ddts/bin)
endif
```

- The first line, beginning with the hash mark (#) is a comment line.
- Note the back quote marks (`) inside the double quotes (“).
- The above lines should be entered into the .cshrc file, one per line as shown.

3 Save the changes made to the .cshrc file and exit the editor.

- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor's documentation.

4 At the UNIX prompt on the AIT Sun, type **source $HOME/.cshrc**, press **Return**.

- This command causes the .cshrc file to get run and the new commands in it to be executed.
- Optionally, the user may logout and then log back in. The result will be the same as above.

Table 15.2.1-2 Setting Up the Environment for DDTS - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	vi $HOME/.cshrc	press Return

2	#DDTS NCR Tool path
	if (“ `echo $path | grep /home/ddts/bin`” == “”) then
	    set path=($path /home/ddts/bin)
	endif
		add lines to .cshrc file

3	Save and quit the editor	invoke editor command
4	source $HOME/.cshrc	press Return

15.2.2 Submitting a New Problem Report

This procedure describes how to submit either a science software or an ECS problem report using DDTS. See Section 15.2.1 for how to set up environment for DDTS.

The Activity Checklist table that follows provides an overview of the process for submitting a science software problem report using DDTS. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 15.2.2-1 Submitting a New Problem Report - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Log onto DDTS server.	(I) 15.2.2	
2	SSI&T	Set the DISPLAY environment variable.	(I) 15.2.2	
3	SSI&T	Invoke the DDTS GUI.	(I) 15.2.2	
4	SSI&T	Choose submit a new defect.	(I) 15.2.2	
5	SSI&T	Enter parameters defining the problem.	(I) 15.2.2	
6	SSI&T	Enter a text description of the problem.	(I) 15.2.2	
7	SSI&T	Save problem description and submit the problem report.	(I) 15.2.2	

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then telnet to the DDTS server.

- Alternatively, in any currently available xterm window, spawn a new session: type **xterm &**, press **Return**. Then telnet to the DDTS server.
- The DDTS server should be known, if not, seek SA.

2 At the UNIX prompt on the DDTS server, type **setenv DISPLAY *hostname*:0.0**, press **Return**.

- The ***hostname*** is the name of the machine on which the DDTS GUI is to be displayed, *i.e.* the machine that your are using.
- If the machine your are on is the DDTS server, this step should not be necessary; the variable should be set by default to **:0.0**, meaning the machine that you are on.
- To verify a setting, type **echo $DISPLAY**, press **Return**.

3 At the UNIX prompt on the DDTS server, type **xddts**, press **Return**.

- The DDTS GUI should be displayed with three distinct windows labeled: **PureDDTS 3.2.1**, **Record**, and **Enclosure**.
- If all three windows do not appear, then a default set up does not exist. In this case, execute the following steps (otherwise, go on to step 4):
  1. Click on the **Select** menu and then choose **Select Project, Class & State…**. A GUI labeled **Select Project, Class & State** will be displayed.
  2. Under the label **Which Class**, click on the **Change Class** button and then choose **Release_A**. Either double click on this entry or single click and then click on the **OK** button.
  3. If the **Change Class** button does not appear, click on the bottom of the scroll bar towards the right to reveal it.
  4. Under the label **Which Projects**, using the scroll bar, click on a project relevant to your problem, for example **PDPS_A**.
  5. Under the **State** field, click on the entry **New** to highlight.
  6. Click on the **Save as Default** button (there will be no visible response).
  7. Finally, click on the **OK** button.
  8. The above parameters set will be saved as your default.

4 When entering a new problem report, click on the **File** menu and select **Submit New Record**.

- The GUI labeled **Record** (the middle panel by default) will be refreshed and prompt for input.

5 In the **Record** GUI, enter the prompted information.

- At the prompt **Submit to which class of Projects:**, your default project should be listed. If so, just press **Return**. If not, enter the class and then press **Return**.
- At the prompt **Project name:**, enter a project name and press **Return**. To see list of valid project names, press **Return**, then make a selection in the displayed GUI.
- At the prompt **NCR Title:**, enter a descriptive title of the problem, then press **Return**. Try to make the problem clear from the title. The title may be free text up to 72 characters in length.
- At the prompt **Software:**, enter the name of the software or program involved in the problem being reported, then press **Return**. This may be any string up to 20 characters in length.
- At the prompt **Build Name**, press **Return** to see a list of valid choices. Highlight a selection and click on the **OK** button.
- At the prompt **Site ID**, press **Return** to see a list of valid choices. Highlight a selection and click on the **OK** button.

- At the prompt **Test Case ID**, enter an identification of the test case involved in the problem, then press **Return**. This may be any string.
- At the prompt **Detection Method**, press **Return** to see a list of valid choices. Highlight a selection and click on the **OK** button.
- At the prompt **Detected-In-Phase**, press **Return** to see a list of valid choices. Highlight a selection and click on the **OK** button. The choices may vary depending on Class.
- At the prompt **Problem Severity (1-5)**, enter the severity of the problem being reported, then press **Return**. A severity of 1 is catastrophic; a severity of 5 is considered minor. Enter **?** and press **Return** to see a list of definitions for each severity level.
- At the prompt **Do you want to be notified of changes to this defect**, enter **Y** for yes or **N** for no. If yes, you will receive electronic mail notifications when the status of this problem report has been changed.
- After the above information has been entered, press **Return**. A GUI labeled **Problem** will be displayed.

6 In the GUI labeled **Problem**, enter a description of the problem.
- Enough information should be provided to allow developers to isolate the problem and the description of the problem should be clear enough for SSI&T personnel to understand the nature of the problem.
- The editor provided for entering a problem description is very simple and allows files to be imported and attached to the description.
- Files such as screen dumps may be placed in the directory /home/ddts/bin/save-files/ on the DDTS server. The file name of the file can then be included as a Problem Description enclosure.

7 When the problem description has been completed, click on the **File** menu and select **Save Changes & Dismiss Editor**.
- Enclosures appear as icons in the enclosures window and can be displayed by double clicking on them.
- The submitter's UNIX userid and electronic mail address are automatically saved with the problem report.

Table 15.2.2-2 Submitting a New Problem Report - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	Tools → Xterm	telnet to DDTS server
2	setenv DISPLAY *hostname*:0.0	press Return
3	xddts	press Return
4	File → Submit New Record	(No action)

Table 15.2.2-2 Submitting a New Problem Report - Quick-Step Procedures (cont.)

Step	What to Enter or Select	Action to Take
5	*class* *project name* *NCR title* *software* *build name* *site ID* *test case ID* *detection method* *detected-in-phase* *problem severity* Y | N	press Return press Return press Return press Return press Return press Return press Return press Return press Return press Return press Return
6	text description of problem	(No action)
7	File → Save Changes & Dismiss Editor	(No action)

15.2.3 Changing or Revising a Problem Report

Sometimes you may wish to modify or revise an NCR you have previously submitted

1 If you are already in the project and problem report skip this procedure. After starting as described in 1-3 above, From the **Select** menu select **Class, Project, & State ...**

2 In the window select the NCR you wish to modify. Make the appropriate modifications to the input screen or the enclosure fields.

3 Select Submit & Dismiss

16. Troubleshooting and General Investigation

An important part of SSI&T is verifying that the output files produced at the DAAC are identical (within particular tolerances) to the test output files delivered with the DAPs. A successful comparison is a strong indication that the porting of the science software from the development facility at the SCF to the operational facility at the DAAC has not introduced any errors.

A number of file comparison tools are available during SSI&T via the SSIT Manager GUI or they can be invoked from the UNIX command line. Two tools are available for comparing HDF or HDF-EOS files, one tool for comparing ASCII files, and another tool for assisting in comparing binary files.

It is assumed that the Instrument Team has delivered test output files (produced at their SCF) with which to perform the comparison.

16.1 Examining PGE Log Files

Three log files are produced by PGEs during runtime: the Status log, User Log, and the Report log. These log files are written by the SDP Toolkit and by the science software using the Toolkit's Status Message Facility (SMF). The location of these log files is specified in the Process Control File (PCF). When the PGE is built and run with the SCF version of the SDP Toolkit, the location and file names of the log files can be set as desired. When the PGE is built with the DAAC version of the SDP Toolkit and run within the PDPS, the location and file names of the log files is set by the system in the instantiated PCF.

The Status log file captures all error and status information. The User log file captures a subset of messages which are more informational. The Report log file captures arbitrary message strings sent by the PGE.

Section 16.1.1 describes how to examine log files produced by PGEs that have been built with the SCF version of the SDP Toolkit and run from the command line.

Sections 16.1.2 describes how to examine log files (within the Production History) produced by PGEs that have been built with the DAAC version of the SDP Toolkit and run within the PDPS.

16.1.1 Log Files From PGEs Run Outside of the PDPS

When the PGE is run outside of the PDPS, the PCF specifies the location and file names of the log files produced. This procedure describes how to locate that information from the PCF and use it to examine the log files.

The Activity Checklist table that follows provides an overview of the process for examining log files from PGEs run outside of the PDPS. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 16.1.1-1 Log Files From PGEs Run Outside of the PDPS - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the location of the PCF.	(I) 16.1.1	
2	SSI&T	Invoke editor with the PCF.	(I) 16.1.1	
3	SSI&T	Note path and file names of log files in the PCF.	(I) 16.1.1	
4	SSI&T	Examine the Status log file.	(I) 16.1.1	
5	SSI&T	Examine the User log file.	(I) 16.1.1	
6	SSI&T	Examine the Report log file.	(I) 16.1.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PGE has been successfully built with the SCF version of the SDP Toolkit (Section 9.4).
2. The PGE's PCF has been updated properly for the DAAC environment (Section 9.1).

To examine log files from a PGE run outside of the PDPS, execute the procedure steps that follow:

1 At the UNIX prompt on an AIT Sun or on the SPR SGI, type **cd *PCFpathname***, press **Return**.
- The ***PCFpathname*** is the full path name to the location of the PCF used by the PGE for which log files are to be examined.

2 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **vi *PCFfilename***, press **Return**.
- The ***PCFfilename*** is the file name of the PCF used by the PGE for which log files are to be examined.
- This brings up the file named ***PCFfilename*** in the *vi* editor.
- Any text editor may be used such as *emacs*. For example, **emacs MOD35.pcf**, press **Return**.

3 In the editor, search for logical IDs (beginning in the first column) **10100**, **10101**, and **10102**. These are the PCF entries for the Status log, User log, and Report log respectively. For each, note the file names in field 2 and the path names in field 3. Then quit the editor.

- If field 3 is blank, then the location is given by the default location specified in a line above the entries beginning with the "!" character.

4 At the UNIX prompt on the SPR SGI, type **vi *StatusLogPathname*/*filename***, press **Return**.

- The ***StatusLogPathname*/*filename*** is the full path name and file name of the Status log file noted in step 3 associated with PCF logical ID 10100. When finished, quit the editor.
- Note any error or warning messages in file.
- Any text editor may be used such as *emacs*. For example, **emacs /PGE/MOD35/LogStatus**, press **Return**.

5 At the UNIX prompt on the SPR SGI, type **vi *UserLogPathname*/*filename***, press **Return**.

- The ***UserLogPathname*/*filename*** is the full path name and file name of the Status log file noted in step 3 associated with PCF logical ID 10101. When finished, quit the editor.
- Note any error or warning messages in file.
- Any text editor may be used such as *emacs*. For example, **emacs /PGE/MOD35/LogUser**, press **Return**.

6 At the UNIX prompt on the SPR SGI, type **vi *ReportLogPathname*/*filename***, press **Return**.

- The ***ReportLogPathname*/*filename*** is the full path name and file name of the Status log file noted in step 3 associated with PCF logical ID 10102. When finished, quit the editor.
- Note any anomalous messages in file.
- Any text editor may be used such as *emacs*. For example, **/PGE/MOD35/LogReport**, press **Return**.

Table 16.1.1-2 Log Files From PGEs Run Outside of the PDPS - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *PCFpathname*	press Return
2	vi *PCFfilename*	press Return
3	(No entry)	note path and file names for IDs 10100, 10101, and 10102
4	vi *StatusLogPathname*/*filename*	press Return, review file, then quit editor
5	vi *UserLogPathname*/*filename*	press Return, review file, then quit editor
6	vi *ReportLogPathname*/*filename*	press Return, review file, then quit editor

16.1.2 Production History Log Files From PGEs Run Within the PDPS

The Production History (PH) is created during PGE execution within the PDPS and then Inserted into the IMF Data Server upon PGE completion. The PH is a UNIX tar file that includes the PGE log files.

See Section 16.2 for listing of other files contained in the PH.

The Activity Checklist table that follows provides an overview of the process for examining the Production History log files from PGEs run within the PDPS. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 16.1.2-1 Production History Log Files From PGEs Run Within the PDPS - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the IMF Data Server Production History archive.	(I) 16.1.2	
2	SSI&T	List out the Production History tar files in the archive and locate the one of interest.	(I) 16.1.2	
3	SSI&T	Copy the desired Production History tar file to a working directory.	(I) 16.1.2	
4	SSI&T	Change directories to the working directory.	(I) 16.1.2	
5	SSI&T	Untar the Production History tar file.	(I) 16.1.2	
6	SSI&T	Examine the Status log file.	(I) 16.1.2	
7	SSI&T	Examine the User log file.	(I) 16.1.2	
8	SSI&T	Examine the Report log file.	(I) 16.1.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PDPS IMF archive configuration area is properly set up.
2. The environment variable DataServer contains the full path name to the archive.

To examine the Production History PGE log files, execute the procedure steps that follow:

1 At the UNIX prompt on an AIT Sun or on the SPR SGI, type **cd $DataServer/PH**, press **Return**.

- The **$DataServer** is an environment variable containing the full path name of the IMF Data Server archive and **PH** is a subdirectory under **$DataServer** containing the Production History tar files.
- For example, type **cd $DataServer/PH**, press **Return**.

2 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **ls -al**, press **Return**.

- A list of the current contents will be displayed. These will be Production History tar files.
- The file names of PH files are named *PGEname#versionMMDDYYhhmm*_runtime.tar_*UR* where *PGEname* is replaced by the PGE name, *version* is replaced by the PGE version, *MMDDYY* is replaced by the Insert date in the month-day-year format, *hhmm* is replaced by the Insert time in the hours-minutes format, and *UR* is replaced by the Universal Reference. For example, the PH file name for version 2.1 of a SAGE III PGE named sage_1t Inserted on December 14, 1999 at 2:00pm could be: sage_1t#2.11214991400_runtime.tar_YAAa005Li_19991214135815.
- Look for the PH of interest.

3 At a UNIX prompt on the AIT Sun or on the SPR SGI, type **cp *PHtarFilename WorkingPathname***, press **Return**.

- The ***PHtarFilename*** is the file name of the Production History tar file.
- The ***WorkingPathname*** is the full path name to some working directory in which the Production History tar file is to be placed and examined.

4 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **cd *WorkingPathname***, press **Return**.

- The ***WorkingPathname*** is the full path name to the working directory specified in step 3.

5 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **tar xvf *PHtarFilename***, press **Return**.

- The ***PHtarFilename*** is the file name of the Production History tar file in the working directory.
- This command will untar the Production History tar file, extracting its component files into the current directory.

6 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **vi *StatusLogFilename***, press **Return**.

- The ***StatusLogFilename*** is the file name of the Status log file within the PH. When finished, quit the editor.
- Note any error or warning messages in file.
- Any text editor may be used such as *emacs*. For example, **emacs LogStatus**, press **Return**.

7 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **vi *UserLogFilename***, press **Return**.

- The ***UserLogFilename*** is the file name of the User log file within the PH. When finished, quit the editor.

- Note any error or warning messages in file.
- Any text editor may be used such as *emacs*. For example, **emacs LogUser**, press **Return**.

8 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **vi *ReportLogFilename***, press **Return**.
- The ***ReportLogFilename*** is the file name of the Report log file within the PH. When finished, quit the editor.
- Note any error or warning messages in file.
- Any text editor may be used such as *emacs*. For example, **emacs LogReport**, press **Return**.

Table 16.1.2-2 Production History Log Files From PGEs Run Within the PDPS - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *PHArchivePathname*	press Return
2	ls -al	press Return
3	cp *PHtarFilename WorkingPathname*	press Return
4	cd *WorkingPathname*	press Return
5	tar xvf *PHtarFilename*	press Return
6	vi *StatusLogFilename*	press Return, review file, then quit editor
7	vi *UserLogFilename*	press Return, review file, then quit editor
8	vi *ReportLogFilename*	press Return, review file, then quit editor

16.2 The Production History

The Production History (PH) is a UNIX tar file that is produced and archived for every run of a PGE in the PDPS. Each PH can be uniquely retrieved from the IMF Data Server.

PH files are located in the $DataServer/PH directory and have the following naming convention:

*PGEname#versionMMDDYYhhmm*_runtime.tar_*UR*

where *PGEname* is replaced by the PGE name, *version* is replaced by the PGE version, *MMDDYY* is replaced by the Insert date in the month-day-year format, *hhmm* is replaced by the Insert time in the hours-minutes format, and *UR* is replaced by the Universal Reference.

For example, a PH file name for version 2.1 of a SAGE III PGE named sage_1t Inserted on December 14, 1999 at 2:00pm would be:

sage_1t#2.11214991400_runtime.tar_YAAa005Li_19991214135815.

The Activity Checklist table that follows provides an overview of the process for examining the Production History. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 16.2-1 The Production History - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the IMF Data Server Production History archive.	(I) 16.2	
2	SSI&T	List out the Production History tar files in the archive and locate the one of interest.	(I) 16.2	
3	SSI&T	Copy the desired Production History tar file to a working directory.	(I) 16.2	
4	SSI&T	Change directories to the working directory.	(I) 16.2	
5	SSI&T	Untar the Production History tar file.	(I) 16.2	
6	SSI&T	Examine the various component files.	(I) 16.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PDPS IMF archive configuration area is properly set up.
2. The environment variable DataServer contains the full path name to the archive.

To examine the Production History file, execute the procedure steps that follow:

1 At the UNIX prompt on an AIT Sun or on the SPR SGI, type **cd $DataServer/PH**, press **Return**.

- The **$DataServer** is an environment variable containing the full path name of the IMF Data Server archive and **PH** is a subdirectory under **$DataServer** containing the Production History tar files.
- For example, type **cd $DataServer/PH**, press **Return**.

2 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **ls -al**, press **Return**.

- A list of the current contents will be displayed. These will be Production History tar files.
- The file names of PH files are named *PGEname#versionMMDDYYhhmm*_runtime.tar_*UR* where *PGEname* is replaced by the PGE name, *version* is replaced by the PGE version, *MMDDYY* is replaced by the Insert date in the month-day-year format, *hhmm* is replaced by the Insert time in the hours-minutes format, and *UR* is replaced by the Universal Reference. For example, the PH file name for version 2.1 of a SAGE III PGE named sage_1t

Inserted on December 14, 1999 at 2:00pm could be:
sage_1t#2.11214991400_runtime.tar_YAAa005Li_19991214135815.

- Look for the PH of interest.

3 At a UNIX prompt on the AIT Sun or on the SPR SGI, type **cp *PHtarFilename WorkingPathname***, press **Return**.

- The ***PHtarFilename*** is the file name of the Production History tar file.
- The ***WorkingPathname*** is the full path name to some working directory in which the Production History tar file is to be placed and examined.

4 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **cd *WorkingPathname***, press **Return**.

- The ***WorkingPathname*** is the full path name to the working directory specified in step 3.

5 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **tar xvf *PHtarFilename***, press **Return**.

- The ***PHtarFilename*** is the file name of the Production History tar file in the working directory.
- This command will untar the Production History tar file, extracting its component files into the current directory.

6 At the UNIX prompt on the AIT Sun or on the SPR SGI, type **vi *PHcomponentFile***, press **Return**.

- The ***PHcomponentFile*** is the file name of one of the PH component files. The component files are:
  - *PGEname#versionMMDDYYhhmm*.Log - Contains the DPR ID, the actual command used to run the PGE, resource usage information, the PGE exit status, and files used by the PGE.
  - *PGEname#versionMMDDYYhhmm*.Pcf - The actual instantiated PCF used when running the PGE.
  - *PGEname#versionMMDDYYhhmm*.ProdLog - Contains the DPR ID, the PGE ID, and resource usage information (same as in the .Log file).
  - *PGEname#versionMMDDYYhhmm*.Profile - Contains the environment variables defined during the execution of the PGE including the contents of the PATH environment variable.
  - *PGEname#versionMMDDYYhhmm*.TkReport - The Report log file, same as is produced when run outside of the PDPS.
  - *PGEname#versionMMDDYYhhmm*.TkStatus - The Status log file, same as is produced when run outside of the PDPS.
  - *PGEname#versionMMDDYYhhmm*.TkUser - The User log file, same as is produced when run outside of the PDPS.
  - *ESDTmmddyyHHMM*.met - All target MCFs for all Inserts on behalf of the PGE. The *ESDT* is the ESDT ShortName into which the file was Inserted, *mmddyy* is the month, day, and year of the Insert and *HHMM* is the time of the Insert.

Table 16.2-2 The Production History - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *PHArchivePathname*	press Return
2	ls -al	press Return
3	cp *PHtarFilename WorkingPathname*	press Return
4	cd *WorkingPathname*	press Return
5	tar xvf *PHtarFilename*	press Return
6	vi *PHcomponentFile*	press Return, review file, then quit editor

16.3 Examining PDPS-Related Scripts and Message Files

This section describes how users may access files, in addition to the PGE-produced log files, which are created during the execution of a DPR job and which may hold information useful in tracing processing problems.

Some of these files are written by default to directory paths that can only be accessed on either the SGI processor machine or one of the Sun workstations. More detailed descriptions of these files and the conditions under which they are generated will be supplied in future Green Book versions.

16.3.1 Examining AutoSys JIL Scripts

JILxxxxxxxxx is the Job Information Language (JIL) script that defines the DPR job to **AutoSys** and which must be submitted to the **AutoSys** Database before a DPR job can be run. The name of the file created is system-generated and begins with the characters ‘JIL’ followed by nine characters (e.g. JILAAAa0066c).

Sample file content:

```
insert_job: 5251_823122483_1

job_type: command

command:    /usr/ecs/Rel_A/CUSTOM/data/bin/sgi/DpAtExecutionMain
5251_823122483_1

machine: spr1sgigsfc

std_out_file: /home/cboettch/mockpge_msfc/out/dpat_std.out

std_err_file: /home/cboettch/mockpge_msfc/out/dpat_std.err

profile: /usr/ecs/Rel_A/CUSTOM/data/bin/sgi/DpAtRunProfile.sh
```

The Activity Checklist table that follows provides an overview of the process for examining the JILxxxxxxxxx scripts on the AIT Sun. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 16.3.1-1 Examining AutoSys JIL Scripts - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the location of the JILxxxxxxxx script.	(I) 16.3.1	
2	SSI&T	Invoke editor with the JILxxxxxxxx script.	(I) 16.3.1	
3	SSI&T	Examine the JILxxxxxxxx script.	(I) 16.3.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

To examine JILxxxxxxxx scripts on the AIT Sun, execute the procedure steps that follow:

1 At the UNIX prompt on an AIT Sun, type **cd *JILscriptPathname***, press **Return**.
- The ***JILscriptPathname*** is the full path name to the location of the JILxxxxxxxx scripts to be examined.

2 At the UNIX prompt on the AIT Sun, type **vi *JILscriptFilename***, press **Return**.
- The ***JILscriptFilename*** is the file name of the JILxxxxxxxx script to be examined.
- This brings up the file named ***JILscriptFilename*** in the *vi* editor.
- Any text editor may be used such as *emacs*. For example, **emacs *JILscriptFilename***, press **Return**.

Table 16.3.1-2 Examining AutoSys JIL Scripts - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *JILscriptPathname*	press Return
2	vi *JILscriptFilename*	press Return

16.3.2 Examining Application Log Files

Most of the custom code used during SSI&T routinely produce log files. For example, the SSIT Manager produces a log file named DpAtMgr.log and the tool used to Insert SSEPs to the IMF Data Server (DpAtInsertExeTarFile.sh) produces a log file named DpAtInsertExeTarFile.log. These files are placed in the directory in which the tool was executed. If the SSIT Manager is run from the user's home directory, then the log files for each of the associated tools will be found in the user's home directory. Log files are produced at the first invocation of the tools, even if no messages are written to them. During subsequent use of the tools, the associated log files will be appended.

Log files are generally named according to the convention:

ApplicationName.log

where *ApplicationName* is replaced with the name of the tool's executable binary. For tools that are shell scripts (*e.g.* .sh files), the shell name is left out of the log file name. For example, the tool DpAtInsertStaticFile.sh produces a log file named DpAtInsertStaticFile.log and not DpAtInsertStaticFile.sh.log.

The Activity Checklist table that follows provides an overview of the process for examining the application log files. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 16.3.2-1 Examining Application Log Files - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Change directories to the location of the application log files.	(I) 16.3.2	
2	SSI&T	Invoke text editor/viewer with the application log file.	(I) 16.3.2	
3	SSI&T	When finished, exit the editor/viewer.	(I) 16.3.2	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The particular application makes use of and produces log files.
2. The location of the application log file is known.
3. The name of the application producing the log file is known.

To examine the application log files, execute the procedure steps that follow:

1 At the UNIX prompt on an AIT Sun or on the SPR SGI, type **cd *LogFilesPathname***, press **Return**.

- The ***LogFilesPathname*** is the full path name to the location of a particular application log file. This is typically the directory from which the SSIT Manager or other tool is run.
- For example, type **cd $HOME/ceres**, press **Return**.

2 At a UNIX prompt on an AIT Sun or on the SPR SGI, type **vi *ApplicationName*.log**, press **Return**

- This ***ApplicationName*.log** is the file name of the log file to examine.
- Any text editor may be used such as *emacs*. For example, **emacs QAMonitor.log**, press **Return**.

3 When finished, exit the editor.

- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor's documentation.

Table 16.3.2-2 Examining Application Log Files - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	cd *LogFilesPathname*	press Return
2	vi *ApplicationName*.log	press Return
3	:wq	press Return

17. Miscellaneous

17.1 Setting Up Environment for Direct PDPS Database Access

The PDPS database contains many tables containing information about science software running in the production system. The population of some of this information is part of standard SSI&T procedures (for example, see Section 11.1) and no special environment set up is required for these procedures. It may be necessary, however, to gain direct access to the PDPS database from time to time and this procedure describes how to set up the required environment.

The Activity Checklist table that follows provides an overview of the process for setting up the environment for direct PDPS database access. Column one (**Order**) shows the order in which tasks should be accomplished. Column two (**Role**) lists the Role/Manager/Operator responsible for performing the task. Column three (**Task**) provides a brief explanation of the task. Column four (**Section**) provides the Procedure (P) section number or Instruction (I) section number where details for performing the task can be found. Column five (**Complete?**) is used as a checklist to keep track of which task steps have been completed.

Table 17.1-1 Setting Up Environment for Direct PDPS Database Access - Activity Checklist

Order	Role	Task	Section	Complete?
1	SSI&T	Invoke text editor with the .cshrc file.	(I) 17.1	
2	SSI&T	Add lines to the .cshrc file.	(I) 17.1	
3	SSI&T	Save the file and quit the editor.	(I) 17.1	
4	SSI&T	Reinitialize the environment.	(I) 17.1	

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The user has been given read access (at a minimum) for the PDPS database.
2. The C shell or a derivative (*e.g.* T shell) is the current user shell.

To set up an environment for PDPS database access, execute the procedure steps that follow:

1 At a UNIX prompt on an AIT Sun or on the SPR SGI, type **vi $HOME/.cshrc**, press **Return**

- This brings up the file named .cshrc in the *vi* editor.

- Any text editor may be used such as *emacs*. For example, **emacs $HOME/.cshrc**, press **Return**.

2 In the editor add the following lines if not already there:

> **setenv SYBASE /vendor/sybase**
> **setenv SYBASE /vendor/sybase**
> **setenv SYBROOT $SYBASE/sybooks**
> **setenv EBTRC $SYBROOT/sun5m/.ebtrc**
> **setenv DSQUERY *computer-server***
> **set path = ($path $SYBASE $SYBASE/bin $SYBROOT/sun5m/bin)**

- The ***computer-server*** is the name of the server at the local DAAC and should be known. If not known, ask your SA or DBA.
- The above lines should be entered into the .cshrc file, one per line as shown.

3 Save the changes made to the .cshrc file and exit the editor.
- The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq**, press **Return**.
- For other editors, refer to that editor's documentation.

4 At the UNIX prompt on the AIT Sun, type **source $HOME/.cshrc**, press **Return**.
- This will reinitialize the setting in the .cshrc file.
- The environment set up for access to the PDPS database should now be complete.

Table 17.1-2 Setting Up Environment for Direct PDPS Database Access - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	vi $HOME/.cshrc	press Return
2	setenv SYBASE /vendor/sybase setenv SYBASE /vendor/sybase setenv SYBROOT $SYBASE/sybooks setenv EBTRC $SYBROOT/sun5m/.ebtrc setenv DSQUERY *computer-server* set path = ($path $SYBASE $SYBASE/bin $SYBROOT/sun5m/bin)	add lines to .cshrc file
3	Save and quit the editor	invoke editor command
4	source $HOME/.cshrc	press Return

Appendix A. Metadata Processing in the IMF Data Server

Appendix A. Metadata Processing in the IMF Data Server

T. Atwater

This paper explains how IMF Data Server metadata is processed in the Pre-Release B Testbed.

Each time a file is inserted to the IMF Data Server, it is accompanied by an ASCII metadata file. This file is normally named *filename*.met, where *filename* is the name of the data file being inserted. The file is identical in format to the ASCII metadata file generated as output to a PGE for each output product. In the IMF Data Server, this metadata remains in the archive as a flat file, and may be Inspected and Queried.

A.1 Metadata Checks

IMF Testbed code performs checks on the ASCII metadata file at Insert time. If any of the following checks fail, the Insert will not occur. Only INVENTORY (GRANULE) metadata is checked.

1. All attributes with MANDATORY = "TRUE" in the ESDT descriptor must appear in the ASCII metadata file.
2. Product-specific attributes (i.e., those in group INFORMATIONCONTENT) in the ASCII metadata file must have OBJECTs corresponding to their PARAMETERNAMEs in the ESDT descriptor.
3. ASCII metadata file attributes RANGEBEGINNINGDATE and RANGEENDINGDATE must be of the form YYYY-MM-DD, and must be valid dates. Attributes RANGEBEGINNINGTIME and RANGEENDINGTIME must be of the form HH:MM:SS.FFFFF…, and must be valid times. All must be ODL string values (be surrounded by ""). RANGEBEGINNINGDATE and RANGEBEGINNINGTIME must be earlier than RANGEENDINGDATE and RANGEENDINGTIME.
4. ASCII metadata file attribute SIZEMBECSGRANULE must be an ODL floating point value (not be surrounded by "" and must contain a decimal point).
5. ASCII metadata file attribute VERSIONID must be an ODL string value (be surrounded by "").
6. All other attributes must have a corresponding OBJECT in either the ESDT descriptor or in the Toolkit Template MCF file (see Appendix). If it is in the ESDT descriptor, it must also be in the Toolkit Template MCF. The attribute value must be of the ODL data type specified in the Toolkit Template MCF.

A.2 Query/Inspect

Currently all attributes in the ASCII metadata file may be queried or inspected. Exception: Inspect of Product-Specific Attributes is currently not supported.

(Query is intended for collection-level attributes, and Inspect for granule-level attributes.)

A.3 Update Metadata

Any INVENTORYMETADATA attribute in the ASCII metadata file may be updated.

A.4 Notes

Attributes may have only one value. Multi-valued attributes are currently not supported in the Testbed.

A.5 SDP Toolkit Template MCF

This file is mainly used to determine the correct ODL data type (string, integer or floating point) of a given attribute. It is also used as a backup to the descriptor: if an ASCII metadata file contains an attribute that is not in the descriptor, then it must be in the Template MCF.

The version of this file used in the Testbed is a modified version of that delivered with the May 1996 Toolkit. Due to a mix-up this version was delivered in softcopy to the Instrument Teams with the Nov. 1996 Toolkit delivery also – the Nov. 1996 Toolkit Template MCF was not delivered in softcopy (though it was listed in the Nov. 1996 Toolkit Users Guide). Because the ITs were probably mostly using the May 1996 version, we decided to use it also.

The May 1996 Toolkit Template MCF is file /ecs/formal/TOOLKIT/runtime/MCF. Listing A-1 below is the version modified to work with the Testbed IMF Data Server. Changes from the May 1996 Toolkit version are listed in comments.

To get a softcopy of the Testbed version, type this at the UNIX command line:

```
unix% /tools/bin/tar zxvf /ecs/formal/PDPS/REL_A/DPS/testbed/archive.tar.gz archive/.MCF
```

The file will be named *.MCF* (note the ".") and be located in subdirectory *archive* under the current directory.

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed

```
GROUP = INVENTORYMETADATA
GROUPTYPE = MASTERGROUP
/* 3-Feb-97 TWA Added QAGranuleUR so IMF type checking possible */
    OBJECT = QAGranuleUR
        Data_Location  = "PGE"
      TYPE = "STRING"
      NUM_VAL = 1
      Mandatory = "TRUE"
    END_OBJECT = QAGranuleUR

/* 3-Feb-97 SHY Added for IMF */

    OBJECT = FailedPGEName
        Data_Location = "PGE"
        TYPE = "STRING"
      NUM_VAL = 1
        Mandatory = "TRUE"
    END_OBJECT = FailedPGEName

    OBJECT = FailedPGEVersion
        Data_Location = "PGE"
        TYPE = "DOUBLE"
      NUM_VAL = 1
        Mandatory = "TRUE"
    END_OBJECT = FailedPGEVersion

    OBJECT = FailedPGEInsertDate
        Data_Location = "PGE"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
    TYPE = "STRING"
  NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = FailedPGEInsertDate

OBJECT = FailedPGEInsertTime
    Data_Location = "PGE"
    TYPE = "STRING"
  NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = FailedPGEInsertTime

OBJECT = ExePGEName
    Mandatory = "TRUE"
    Data_Location = "PGE"
    TYPE = "STRING"
  NUM_VAL = 1
END_OBJECT = ExePGEName

OBJECT = ExePGESSWVersion
    Mandatory = "TRUE"
    Data_Location = "PGE"
    TYPE = "STRING"
  NUM_VAL = 1
END_OBJECT = ExePGESSWVersion

OBJECT = ExePlatformOS
    Mandatory = "TRUE"
    Data_Location = "PGE"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
    TYPE = "STRING"
  NUM_VAL = 1
END_OBJECT = ExePlatformOS

OBJECT = ExePlatformOSVersion
    Mandatory = "TRUE"
    Data_Location = "PGE"
    TYPE = "STRING"
  NUM_VAL = 1
END_OBJECT = ExePlatformOSVersion

OBJECT = ExeInsertDate
    Mandatory = "TRUE"
    Data_Location = "PGE"
    TYPE = "STRING"
  NUM_VAL = 1
END_OBJECT = ExeInsertDate

OBJECT = ExeInsertTime
    Mandatory = "TRUE"
    Data_Location = "PGE"
    TYPE = "STRING"
  NUM_VAL = 1
END_OBJECT = ExeInsertTime

OBJECT = PlatformShortName
    Data_Location = "MCF"
    TYPE = "STRING"
    NUM_VAL = 1
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
    Mandatory  = "TRUE"
END_OBJECT = PlatformShortName


OBJECT = DataSource
    Data_Location = "MCF"
    TYPE = "STRING"
    NUM_VAL = 1
    Mandatory  = "TRUE"
END_OBJECT = DataSource


OBJECT = NumberOfOrbits
    Data_Location = "PGE"
    TYPE = "INTEGER"
    NUM_VAL = 1
    Mandatory  = "TRUE"
END_OBJECT = NumberOfOrbits


OBJECT = OrbitNumber
    Data_Location = "PGE"
    TYPE = "INTEGER"
    CLASS = "M"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = OrbitNumber


OBJECT = AscendingNodeCrossingDate
    Data_Location = "PGE"
    TYPE = "STRING"
    CLASS = "M"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = AscendingNodeCrossingDate

OBJECT = AscendingNodeCrossingTime
    Data_Location = "PGE"
    TYPE = "STRING"
    CLASS = "M"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = AscendingNodeCrossingTime

OBJECT = EquatorCrossingDate
    Data_Location = "PGE"
    TYPE = "STRING"
    CLASS = "M"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = EquatorCrossingDate

OBJECT = EquatorCrossingTime
    Data_Location = "PGE"
    TYPE = "STRING"
    CLASS = "M"
    NUM_VAL = 1
    Mandatory = "TRUE"
END_OBJECT = EquatorCrossingTime

OBJECT = EquatorCrossingLongitude
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
        Data_Location = "PGE"
        TYPE = "DOUBLE"
        CLASS = "M"
        NUM_VAL = 1
        Mandatory = "TRUE"
    END_OBJECT = EquatorCrossingLongitude

/* *********Associate granule with collection *************** *****/

/* ******IMPORTANT - wherever xxxx or XXX or TBC is used */
/* within this MCF, these are values which the USER will */
/* ultimately fill in themselves - the MCF will not function with */
/* these original values!!!!! *********************************** */
/* IMPORTANT - any text comment in the MCF MUST be enclosed */
/* per line with /* and */.  Unlike C where a block of text may */
/* be enclosed */

/* All attributes EXCEPT those at the highest level associated */
/* with ECSDataGranule (DID 31ll) must be grouped together */
/* The groups MUST relate to the CLASS names within DID 311 */
/*  Container Objects, within a group, must duplicate the group */
/* name and append Container */

/*  IMPORTANT The following Groups are Self Describing */
/* Parameter, SensorCharacteristic, VerticalSpatialDomain */
/* PlatformCharacteristic, Locality, RegularPeriodic and */
/* AdditionalAttribute */
/* They all have values which are self typed - these values */
/* will be set up in the descriptor, prior to MCF generation */
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
/* For MCF purposes, the values of all of these attributes */
/* will be of type STRING.  The database will retain the correct */
/* types.  The PGE/user MUST convert the value they are */
/* assinging to type string, before calling PGS_MET_SET */
/* and setting the value of the attribute.  */

OBJECT = ShortName            /* Name of attribute */
      Data_Location  = "MCF"        /* Set in MCF */
      Value = "xxxx"                /* The value replaces xxxx */
      TYPE = "STRING"
      NUM_VAL = 1
      Mandatory  = "TRUE"
END_OBJECT = ShortName

/*  these ReprocessingStatus values are mandatory */

OBJECT = ReprocessingActual
      Data_Location  = "MCF"
      Value = "xxxx"
      TYPE = "STRING"
      NUM_VAL = n
      Mandatory = "TRUE"
END_OBJECT = ReprocessingActual

OBJECT = ReprocessingPlanned
      Data_Location  = "MCF"
      Value = "xxxx"
      TYPE = "STRING"
      NUM_VAL = n
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
      Mandatory = "TRUE"
END_OBJECT = ReprocessingPlanned


/* the size of the granule may be set once in the collection for */
/* all granule OR calculated routinely per granule (in which case */
/* the Data_location = PGE); which ever is most useful */


/* 15-Jan-97 TWA: Changed from INTEGER to DOUBLE to */
/* support current Dsrv code */


OBJECT = SizeMBECSDataGranule
      Data_Location  = "PGE"
      TYPE = "DOUBLE"
      NUM_VAL = 1
      Mandatory  = "TRUE"
END_OBJECT = SizeMBECSDataGranule



/* There are various ways of expressing spatial coverage.  */
/* ECS requires that the following be supplied. */


/* BoundingRectangle                (mandatory if applicable) */
/* plus none or more of... */
/* Point */
/* GPolygonContainer */
/* Circle */
/* OrbitCalculatedSpatialDomain */


/* Each of these is a compound of several attributes as specified */
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
/* below    */


/* BoundingRectangle set; these must occur only once for each */

/* granule */


GROUP = BoundingRectangle


      OBJECT = EastBoundingCoordinate

            Data_Location = "PGE"

            TYPE = "DOUBLE"

            NUM_VAL = 1

            Mandatory = "TRUE"

      END_OBJECT = EastBoundingCoordinate


      OBJECT = WestBoundingCoordinate

            Data_Location     = "PGE"

            TYPE = "DOUBLE"

            NUM_VAL =  1

            Mandatory = "TRUE"

      END_OBJECT = WestBoundingCoordinate


      OBJECT = NorthBoundingCoordinate

            Data_Location = "PGE"

            TYPE = "DOUBLE"

            NUM_VAL = 1

            Mandatory = "TRUE"

      END_OBJECT = NorthBoundingCoordinate


      OBJECT = SouthBoundingCoordinate
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            Data_Location = "PGE"
            TYPE = "DOUBLE"
            NUM_VAL = 1
            Mandatory  = "TRUE"
      END_OBJECT = SouthBoundingCoordinate

END_GROUP = BoundingRectangle

/*  there may be many GRingContainer object sets per */
/* GRing group i.e. many grings in each granule */

/* GRings must be sequenced i.e. sequence number in a clockwise */
/* manner, the area to the right of the direction of travel being the */
/*  enclosed area */

GROUP = GRing

OBJECT = GRingContainer

      Data_Location= "NONE" /* necessary to i.d. a non-functional */
                              /* container object */
      CLASS = "M"
      Mandatory = "TRUE"

      OBJECT = ExclusionGRingFlag
            Data_Location= "PGE"
            CLASS = "M"
            TYPE = "STRING"
            NUM_VAL = n
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            Mandatory = TBC
      END_OBJECT = ExclusionGRingFlag

/* for each of the following objects there are at least 3 elements */
/* in each array */

      OBJECT = GRingPointLatitude
            Data_Location = "PGE"
            CLASS = "M"
            TYPE = "DOUBLE"
            NUM_VAL = n       /* an array  of max size n */
                                    /* where n is at least 3*/
            Mandatory = TBC
      END_OBJECT = GRingPointLatitude

      OBJECT = GRingPointLongitude
            Data_Location = "PGE"
            CLASS = "M"
            TYPE = "DOUBLE"
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = GRingPointLongitude

      OBJECT = GRingPointSequenceNo
            Data_Location  = "PGE"
            CLASS = "M"
            TYPE = "STRING"
            NUM_VAL = n
            Mandatory = TBC
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
      END_OBJECT = GRingPointSequenceNo


END_OBJECT = GRingContainer

END_GROUP = GRing


/* granule may be described using one circle. */


GROUP = Circle


      OBJECT = CenterLatitude

            Data_Location  = "PGE"

            TYPE = "DOUBLE"

            NUM_VAL =  1

            Mandatory = TBC

      END_OBJECT = CenterLatitude


      OBJECT = CenterLongitude

            Data_Location = "PGE"

            TYPE = "DOUBLE"

            NUM_VAL = 1

            Mandatory = TBC

      END_OBJECT = CenterLongitude


      OBJECT = RadiusValue    /* assumed to have the type */

                              /* of RadiusUnits */

            Data_Location = "PGE"

            TYPE = "DOUBLE"

            NUM_VAL = 1

            Mandatory = TBC
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
      END_OBJECT = RadiusValue

END_GROUP = Circle

/* granule may be described using one point. ! */

GROUP = Point

      OBJECT = PointLatitude
            Data_Location = "PGE"
            TYPE = "DOUBLE"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = PointLatitude

      OBJECT = PointLongitude
            Data_Location = "PGE"
            TYPE = "DOUBLE"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = PointLongitude

END_GROUP = Point

/* when using Grid coordinate systems, ZoneIdentifier may be */
/* repeating (i.e. multiple zones) */

GROUP = ZoneIdentifierClass
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
      OBJECT = ZoneIdentifier
      Data_Location = "PGE"
            TYPE = "INTEGER"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = ZoneIdentifier

END_GROUP = ZoneIdentifierClass

/* when using locality text labels - may be repeating;  */
/*  i.e. many text labels per granule */

GROUP = GranuleLocality

      OBJECT = LocalityValue
            Data_Location = "PGE"
            TYPE = "STRING"
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = LocalityValue

END_GROUP = GranuleLocality

/* Location information for vertical product data, at release A*/
/*  only one vertical value per granule is expected. The TYPE is */
/*  variable dependent on the VerticalSpatialDomainType */

GROUP = VerticalSpatialDomain
      OBJECT = VerticalSpatialDomainValue
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            Data_Location = "PGE"
            TYPE = "STRING"
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = VerticalSpatialDomainValue

END_GROUP = VerticalSpatialDomain

/* orbit related - especially useful for level 0 data */

GROUP = OrbitCalculatedSpatialDomain

      OBJECT = OrbitNumber
            Data_Location = "PGE"
            TYPE = "INTEGER"
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = OrbitNumber

      OBJECT = StartOrbitNumber
            Data_Location = "PGE"
            TYPE = "INTEGER"
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = StartOrbitNumber

      OBJECT = StopOrbitNumber
      Data_Location = "PGE"
            TYPE = "INTEGER"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = StopOrbitNumber

      OBJECT = LongitudeOfEquatorCrossing
            Data_Location = "PGE"
            TYPE = "DOUBLE"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = LongitudeOfEquatorCrossing

      OBJECT = DateOfEquatorCrossing
            Data_Location = "PGE"
            TYPE = "DATETIME"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = DateOfEquatorCrossing

      OBJECT = TimeOfEquatorCrossing
            Data_Location = "PGE"
            TYPE = "DATETIME"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = TimeOfEquatorCrossing


END_GROUP = OrbitCalculatedSpatialDomain

/*  There are several ways to express temporal extent at the */
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
/* granule level.  These are RangeDateTime or SingleDateTime */

/* each of which is a compound.  Repeating sets of time measures */

/* can be placed in the ARCHIVEDMETADATA master group*/


/* range time */


GROUP = RangeDateTime


      OBJECT = RangeBeginningDateTime

            Data_Location = "PGE"

            TYPE = "DATETIME"

            NUM_VAL = 1

            Mandatory = TBC

      END_OBJECT = RangeBeginningDateTime


      OBJECT = RangeEndingDateTime

            Data_Location = "PGE"

            TYPE = "DATETIME"

            NUM_VAL = 1

            Mandatory = TBC

      END_OBJECT = RangeEndingDateTime


END_GROUP = RangeDateTime


/* SingleDateTime */


GROUP = SingleDateTime


      OBJECT = CalendarDateTime
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            Data_Location = "PGE"
            TYPE = "DATETIME"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = CalendarDateTime

END_GROUP = SingleDateTime

/* Information for data quality rating.  */

/* each of the following QA - quality flags or values may be set */
/*  for the whole granule and-or for science parameters within */
/* each granule.  Only whole granule measures for inventory */
/* purposes are included in this master group, repeating sets */
/* can be placed into the ARCHIVEDMETADATA group using */
/* arrays and if necessary nested group and object and CLASS = "M" */
/*  statements where 2 dimensional sets of values per attribute */
/* are required */

GROUP = QAStats

      OBJECT = QAPercentInterpolatedData
            Data_Location  = "PGE"
            TYPE = "INTEGER"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = QAPercentInterpolatedData

      OBJECT =  QAPercentOutofBoundsData
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            Data_Location  = "PGE"
            TYPE = "INTEGER"
            NUM_VAL = 1
            Mandatory = "TRUE"
      END_OBJECT = QAPercentOutofBoundsData

      OBJECT = QAPercentMissingData
            Data_Location  = "PGE"
            TYPE = "INTEGER"
            NUM_VAL = 1
            Mandatory =  TBC
      END_OBJECT = QAPercentMissingData

END_GROUP = QAStats

GROUP = QACollectionStats

/* 24-Jan-97 TWA added ScienceQualityFlag and OperationalQualityFlag */

      OBJECT = ScienceQualityFlag
            Data_Location = "PGE"
            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = ScienceQualityFlag

      OBJECT = OperationalQualityFlag
            Data_Location = "PGE"
            TYPE = "STRING"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = OperationalQualityFlag

      OBJECT = AutomaticQualityFlag
            Data_Location = "PGE"
            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory = TBC
      END_OBJECT = AutomaticQualityFlag

/* 18-Mar-97 TWA Added QACollectionStats/AutomaticQaFlag */
/* This is necessary to test ASTER L1B BTS Beta version */
/* because ASTER software is using an old Toolkit template MCF, */
/* where the attribute name is AutomaticQaFlag, */
/* whereas in the current Toolkit template MCF, */
/* the same attrinute is named AutomaticQualityFlag. */
/* In order to both preserve ASTER software and to get it */
/* to work in PDPS, we must add this extra object here */
/* (Added to template MCF because this is where IMF software */
/* checks the data type */

      OBJECT = AutomaticQaFlag
            Data_Location = "PGE"
            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory = "FALSE"
      END_OBJECT = AutomaticQaFlag
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
END_GROUP = QACollectionStats



/* these are URs made available to the PGE from the PCF */

/* which must be extracted by the PGE individually */


GROUP = InputGranule


OBJECT = InputPointer

            Data_Location = "PGE"

            TYPE = "STRING"

            NUM_VAL = n

            Mandatory = "TRUE"/* every granule has a parent */

      END_OBJECT = InputPointer


END_GROUP = InputGranule


GROUP = AncillaryInputGranule


      OBJECT = AncillaryInputPointer

            Data_Location = "PGE"

            TYPE = "STRING"

            NUM_VAL = n

            Mandatory = TBC

      END_OBJECT = AncillaryInputPointer


END_GROUP = AncillaryInputGranule


GROUP = OrbitParametersGranule
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
      OBJECT = OrbitParametersPointer
            Data_Location = "PGE"
            TYPE = "STRING"
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = OrbitParametersPointer

END_GROUP = OrbitParametersGranule

GROUP = InformationContent

      OBJECT = InformationContentContainer

      CLASS = "M"             /* counter for each additional */
                                    /* attribute */
      Mandatory = "TRUE"
      Data_Location = "NONE"

/* EITHER AdditionalAttributeName or ParameterName is used */
/* to describe a ParameterValue if a value is appropriate */



/* for situations where geophysical parameter */
/*  content varies with the granule.  The TYPE is Parameter */
/*  or AdditionalAttribute */
/* DataType (i.e. user defined).  This group object construct */
/* allows for 2-dimensional  values sets; i.e. where each */
/* parameter has many values per granule.  */
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
      OBJECT = ParameterName
            Data_Location = "PGE"
            TYPE = "STRING"
            CLASS = "M"
            NUM_VAL = 1
            Mandatory =  FALSE
       END_OBJECT = ParameterName

/* for non-core attributes which are not  geophysical parameter */
/*  content.  The TYPE is */
/* AdditionalAttribute DataType (i.e. user defined).  */
/* This group object construct */
/* allows for 2-dimensional  values sets; i.e. where each */
/* additional attribute has many values per granule.  */
/* The value of additional attribute is ParameterValue */

      OBJECT = AdditionalAttributeName
            Data_Location = "PGE"
            TYPE = "STRING"
            CLASS = "M"
            NUM_VAL = 1
            Mandatory =  FALSE
       END_OBJECT = AdditionalAttributeName

/* value may or may not be appropriate to describe the */
/* AdditionalAttributeName or  ParameterName */

      OBJECT =  ParameterValue
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            Data_Location  = "PGE"
            CLASS = "M"
            TYPE = "STRING"
            NUM_VAL =  n
            Mandatory =  FALSE
      END_OBJECT =  ParameterValue

END_OBJECT = InformationContentContainer

END_GROUP = InformationContent

/* these sensor characteristics change relatively rapidly and */
/* need to be store in a database.  The values vary according */
/* to data in the telemetry stream and are processed */
/* automatically.  They are set up in the same way as */
/* Parameter values.   SensorCharacteristicValue has */
/* type SensorCharacteristictype. This group/object construct */
/* allows for 2-dimensional  values sets; i.e. where each */
/* SensorCharacteristicValue has many values per granule.  */


GROUP = SensorCharacteristic

OBJECT = SensorCharacteristicContainer

      CLASS = "M" /* allows for many sensorcharacteristics */
                        /* per granule */
      Mandatory = "TRUE"
      Data_Location = "NONE"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
OBJECT = SensorCharacteristicName
      Data_Location = "PGE"
      CLASS = "M"
      TYPE = "STRING"
      NUM_VAL = 1
      Mandatory = "TRUE"
 END_OBJECT = SensorCharacteristicName

OBJECT =  SensorCharacteristicValue
      Data_Location  = "PGE"
      CLASS = "M"
      NUM_VAL = 1
      Mandatory =  "FALSE"
      TYPE = "STRING"
END_OBJECT =  SensorCharacteristicValue

OBJECT = SensorShortName
      Data_Location = "PGE"
      CLASS = "M"
      TYPE = "STRING"
      NUM_VAL = 1
      Mandatory = "TRUE"
 END_OBJECT = SensorShortName

OBJECT = InstrumentShortName
      Data_Location = "PGE"
      CLASS = "M"
      TYPE = "STRING"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            NUM_VAL = 1
            Mandatory = "TRUE"
      END_OBJECT = InstrumentShortName

/* Instrument and Sensor names needed to allow the */
/* database to locate the correct charactreristic name */

END_OBJECT = SensorCharacteristicContainer

END_GROUP = SensorCharacteristic

GROUP = OperationModeClass

/* this used to monitor instument mode changes needed */
/* at granule level */

OBJECT = OperationMode
      Data_Location = "PGE"
      TYPE = "STRING"
      NUM_VAL = n
      Mandatory = TBC
END_OBJECT = OperationMode

END_GROUP = OperationModeClass

/* end of master group */

END_GROUP = INVENTORYMETADATA
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
GROUP = ARCHIVEDMETADATA

GROUPTYPE = MASTERGROUP

/* this master group may contain any core attributes group */
/* plus product specific attributes.  Both may be single */
/* attributes or repeating.  In the latter case, arrays plus the */
/* Group, Object and CLASS = "M" construct shown above should*/
/*  be used. */

/* This group should be written using MET_WRITE. */
/* These attributes are NOT parsed into the inventory and */
/* are therefore NOT SEARCHABLE */

/* the following are core attributes not required to be searched */
/*  , but which may be mandatory or just useful - according to */
/* choices made in appendix B. */

/* these attribute needed in browse granules */

/* Shortname +                                  same as in granule */
/* BrowseDescription + */
/* BrowseSize + */
/* SpatialDomainContainer +                        same as in granule */

/* [RangeDateTime | SingleDateTime] +       same as in granule */

GROUP = Browse
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
OBJECT =  BrowseSize
      Data_Location  = "PGE" /* may also be MCF (static) provided */
                        /*  if not prone to change */
      Value = xxxx                /* value only present if Data_Location */
                        /* equal to MCF */
      TYPE = "DOUBLE"
      NUM_VAL = 1
      Mandatory = TBC
END_OBJECT =  BrowseSize

OBJECT =  BrowseDescription
      Data_Location  = "MCF"
      Value = " xxxx "
      NUM_VAL = 1
      TYPE = "STRING"
      Mandatory = TBC
END_OBJECT =  BrowseDescription

END_GROUP = Browse

GROUP = Review

      OBJECT = ReviewContainer

      Data_Location = "MCF"
      CLASS = "M"
      Mandatory = "TRUE"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
   OBJECT = ScienceReviewDate
         Data_Location = "MCF"
         TYPE = "DATETIME"
         CLASS = "M"
         NUM_VAL = n
         Mandatory = TBC
   END_OBJECT = ScienceReviewDate

   OBJECT = ScienceReviewStatus
         Data_Location = "MCF"
         TYPE = "INTEGER"
         CLASS = "M"
         NUM_VAL = n
         Mandatory = TBC
   END_OBJECT = ScienceReviewStatus

   OBJECT = FutureReviewDate
         Data_Location = "MCF"
         TYPE = "DATETIME"
         CLASS = "M"
         NUM_VAL = n
         Mandatory = TBC
   END_OBJECT = FutureReviewDate

   END_OBJECT = ReviewContainer

END_GROUP = Review
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
/* the following are core attributes found in the */

/* INVENTORYMETADATA master group but duplicated in this */

/* master group in order to contain additional attribute values */

/* required to be in the product granule but not searchable; for */

/* example, quality measures related to individual geophysical */

/* parameters.  The 2-dimensional formulation using arrays */

/* (NUM_VALS) and CLASS = "M" is used to accomodate many */

/* values in a self describing situation */


GROUP = SensorCharacteristic


OBJECT = SensorCharacteristicContainer


      CLASS = "M"

      Data_Location = "NONE"

      Mandatory = "FALSE"


      OBJECT = SensorCharacteristicName

            Data_Location = "PGE"

            CLASS = "M"

            TYPE = "STRING"

            NUM_VAL = 1

            Mandatory = "FALSE"

       END_OBJECT = SensorCharacteristicName


      OBJECT =  SensorCharacteristicValue

            Data_Location  = "PGE"

            CLASS = "M"

            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory =  "FALSE"
      END_OBJECT =  SensorCharacteristicValue

      OBJECT = SensorShortName
            Data_Location = "PGE"
            CLASS = "M"
            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory =  "FALSE"
       END_OBJECT = SensorShortName

      OBJECT = InstrumentShortName
            Data_Location = "PGE"
            CLASS = "M"
            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory =  "FALSE"
       END_OBJECT = InstrumentShortName

END_OBJECT = SensorCharacteristicContainer

END_GROUP = SensorCharacteristic



GROUP = InformationContent

      OBJECT = InformationContentContainer
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
      CLASS = "M"             /* counter for each additional */
                                    /* attribute */
      Mandatory = "TRUE"
      Data_Location = "NONE"

/* EITHER AdditionalAttributeName or ParameterName is used */
/* to describe a ParameterValue if a value is appropriate */



/* for situations where geophysical parameter */
/*  content varies with the granule.  The TYPE is Parameter */
/* DataType (i.e. user defined).  This group object construct */
/* allows for 2-dimensional  values sets; i.e. where each */
/* parameter has many values per granule.  */

      OBJECT = ParameterName
            Data_Location = "PGE"
            TYPE = "STRING"
            CLASS = "M"
            NUM_VAL = 1
            Mandatory =  FALSE
       END_OBJECT = ParameterName

/* for non-core attributes which are not  geophysical parameter */
/*  content.  The TYPE is */
/* AdditionalAttribute DataType (i.e. user defined).  */
/* This group object construct */
/* allows for 2-dimensional  values sets; i.e. where each */
/* additional attribute has many values per granule.  */
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
/* The value of additional attribute is ParameterValue */


      OBJECT = AdditionalAttributeName
            Data_Location = "PGE"
            TYPE = "STRING"
            CLASS = "M"
            NUM_VAL = 1
            Mandatory =  FALSE
       END_OBJECT = AdditionalAttributeName


/* value may or may not be appropriate to describe the */
/* AdditionalAttributeName or  ParameterName */


      OBJECT =  ParameterValue
            Data_Location  = "PGE"
            CLASS = "M"
            TYPE = "STRING"
            NUM_VAL =  n
            Mandatory =  FALSE
      END_OBJECT =  ParameterValue


END_OBJECT = InformationContentContainer


END_GROUP = InformationContent


/* Information for data quality rating.  */


/* each of the following QA - quality flags or values may be set */
/* for the whole granule and-or for science parameters within */
/* each granule. */
GROUP = QAStats


      OBJECT = QAStatsContainer


      CLASS = "M"             /* counter for each parameter */
      Data_Location = "NONE"
      Mandatory = "FALSE"



      OBJECT = QAPercentInterpolatedData
            Data_Location  = "PGE"
            CLASS = "M"
            TYPE = "INTEGER"
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = QAPercentInterpolatedData

      OBJECT =  QAPercentOutofBoundsData
            Data_Location  = "PGE"
            CLASS = "M"
            TYPE = "INTEGER"
            NUM_VAL = n
            Mandatory = "TRUE"
      END_OBJECT = QAPercentOutofBoundsData

      OBJECT = QAPercentMissingData
            Data_Location = "PGE"
            CLASS = "M"
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            TYPE = "INTEGER"
            NUM_VAL = n
            Mandatory = TBC
      END_OBJECT = QAPercentMissingData

/* parameter name may be required to link the QA measures to */
/* a particular parameter in the granule */

      OBJECT = ParameterName
            Data_Location = "PGE"
            CLASS = "M"
            TYPE = "STRING"
            NUM_VAL = 1
            Mandatory = "FALSE"
       END_OBJECT = ParameterName

END_OBJECT = QAStatsContainer

END_GROUP = QAStats

/* use NUM_VAL here to set multiple values of quality flag for */
/* components of the granule */

GROUP = QACollectionStats

      OBJECT = AutomaticQualityFlag
            Data_Location = "PGE"
            TYPE = "STRING"
            NUM_VAL = n
```

Listing A-1 - SDP Toolkit Template MCF in the Pre-Release B Testbed (cont.)

```
            Mandatory = TBC

      END_OBJECT = AutomaticQualityFlag


END_GROUP = QACollectionStats



END_GROUP = ARCHIVEDMETADATA


END
```

This page intentionally left blank.

Appendix B. Sample Checklist File for the SSIT Manager

Appendix B. Sample Checklist File for the SSIT Manager

B.1 Introduction

The SSIT Manager allows a checklist to be displayed in the main window of the GUI. Items representing SSI&T activities in the checklist can be marked as “done” and comments can be associated with each item as the activities are completed. Refer to Section 7 for procedures on setting up and using the SSIT Manager.

B.2 Sample

Listing B.2-1 contains a complete checklist file that may be used as is (with some changes in the first few lines) or it may serve as a template for customized checklist files.

All items are in “parameter=value” form. Lines beginning with the hash mark (#) are considered comments and are ignored by the SSIT Manager. Additional comments may be added freely. Line one, however, must contain a comment line (with at least the # in the first position).

Listing B.2-1 - Sample Checklist File

```
#
#     DpAtMgr log initialization file.
#
#     parameter=value pairs
#     DATABASE=path
#     CHECKLIST=checklist_name
#     ITEM=checklist_item_name
#
#
DATABASE=/tmp/sample_database
#
CHECKLIST=SSITsamp
#
ITEM=Verify account setup for SSI&T
ITEM=Acquire and unpack Delivered Algorithm Package (DAP)
ITEM=Inspect and review contents of DAP
ITEM=Create view in ClearCase
ITEM=Run script to insert elements of DAP into ClearCase
ITEM=Create ESDT descriptor files
ITEM=Register ESDTs
ITEM=Perform standards checking on PGE
ITEM=Compile Toolkit SMF message files
ITEM=Compile and link PGE with SCF version of Toolkit
ITEM=Run and profile SCF-built PGE
ITEM=Examine output log files from PGE run
ITEM=Compare PGE run results with test comparison data
```

Listing B.2-1 - Sample Checklist File (cont.)

```
ITEM=View output products

ITEM=Create ESDT PDPS metadata ODL files

ITEM=Create template PGE PDPS metadata ODL file, edit and update
PDPS database

ITEM=Supply operational metadata and complete PGE registration

ITEM=Create Target MCFs for MCF files

ITEM=Create Target MCFs for static data files

ITEM=Create Target MCFs for dynamic data files

ITEM=Insert MCFs to Data Server.

ITEM=Insert static date files to Data Server.

ITEM=Insert dynamic data files to Data Server.

ITEM=Register subscription for test dynamic input and output
files

ITEM=Compile user Toolkit message files

ITEM=Compile and link PGE with DAAC version of Toolkit

ITEM=Create Science Software Executable Package (SSEP)

ITEM=Create Target MCF for SSEP

ITEM=Insert SSEP to Data Server

ITEM=Create a Production Request for single PGE run

ITEM=Plan and schedule the PR.

ITEM=Monitor the PGE in the PDPS

ITEM=Examine log files and other files from Production History
tar file

ITEM=Compare PGE run results with test comparison data

ITEM=View output products
```

Appendix C. Environment Variables Used in SSI&T

Appendix C. Environment Variables Used in SSI&T

C.1 Introduction

Environment Variable	Description	Used By
AUTOSYS	Contains the full path name to the AutoSys/AutoXpert software directory. Nominally, set to $ECS_HOME/COTS/autosys	AutoSys/AutoXpert on PLN Sun (Section 12.4)
AUTOUSER	Contains the full path name to location of user configurations, Event Processor output files, and sound files. Nominally, set to $ECS_HOME/COTS/autotree2	AutoSys/AutoXpert on PLN Sun (Section 12.4)
AUTOSERV	Contains the name of the AutoSys instance. Nominally, set to PLN	AutoSys/AutoXpert on PLN Sun (Section 12.4)
DISPLAY	Contains the machine name and device of the current display. The machine name is the IP address or the fully qualified domain name following by :0.0 (the device).	All GUIs and xterms
DPAT_ESDT_SCIENCE_MD	Contains the full path name to the directory containing the ESDT metadata ODL files for PDPS. Nominally, set to $ECS_HOME/test/data on the AIT Suns.	DpAtPdpsDbUpdateScience.sh on AIT Sun (Section 11.1.2)
DPATMGR_DAT	Contains the full path name to the directory containing data files and scripts used by the SSI&T tools. Nominally, set to $DPATMGR_HOME/data/DPS	

	on the AIT Suns.	
DPATMGR_HOME	Contains the full path name to the directory containing the SSI&T tools. Nominally, set to /data2/CUSTOM on the AIT Suns.	
DPAT_PGE_SCIENCE_MD	Contains the full path name to the directory containing the PGE metadata ODL files for PDPS. Nominally set to $ECS_HOME/test/output	DpAtPdpsDbUpdateScience.sh on AIT Sun (Section 11.1.2)
DSQUERY	Contains the name of the Sybase dataserver	AutoSys/AutoXpert on PLN Sun (Section 12.4)
ECS_HOME	Contains the full path name to the directory containing the ECS Release A software. Nominally, set to /usr/ecs/Rel_A	EOSView on AIT Sun (Sections 14.1, 14.2)
ECS_MODE	Contains the mode in which the system is being used. Nominally, set to ops.	
EOSVIEWHELPDIR	Contains the full path name to the directory containing the files eosview.csc and eosview.dat	EOSView on AIT Sun (Sections 14.1, 14.2)
EOSVIEWDIR	Contains the full path name to the directory containing EOSView. Nominally, set to $ECS_HOME/CUSTOM/bin/ CLS/EOSVIEW	EOSView on AIT Sun (Sections 14.1, 14.2)
FCKCNF	Contains the full path and file name of the FORCHECK configuration file. Nominally, set to /data2/CUSTOM/data/DPS/ fckcnf.ecs	FORCHECK on AIT Sun
IDL_DIR		
IDLPATH		

IDL_PATH		
PGSHOME	Contains the full path name to the SDP Toolkit home directory. For SSIT Manager, it is nominally set to $ECS_HOME/CUSTOM/bin/ daac_toolkit_f77/TOOLKIT on the SPR SGI.	SSIT Manager on AIT Sun (Sections 7.1, 7.2, 7.3) Science Software on SPR SGI (Sections 9.2, 9.3, 9.4, 9.5, 10.1)
PGSMSG	Contains the full path name to the SDP Toolkit message files. Nominally set to $PGSHOME/message on the SPR SGI.	SSIT Manager on AIT Sun (Sections 7.1, 7.2, 7.3) Science Software on SPR SGI (Sections 9.2, 9.3, 9.4, 9.5, 10.1)
PGS_PC_INFO_FILE	Contains the full path and file name of the Process Control File. For SSIT Manager, the default is $ECS_HOME/CUSTOM/ DpAtMgrInternal.pcf.$HOST on the AIT Suns.	SSIT Manager on AIT Sun (Sections 7.1, 7.2, 7.3) Science Software on SPR SGI (Sections 9.2, 9.3, 9.4, 9.5, 10.1)
SYBASE	Contains the full path name to the directory containing the Sybase software. Nominally, set to $ECS_HOME/COTS/sybase	AutoSys/AutoXpert on PLN Sun (Section 12.4)
UIDPATH	Contains the full path name to the directory containing the eosview.uid file which contains descriptions of GUI objects.	EOSView on AIT Sun (Sections 14.1, 14.2)

ECS_HOME = /usr/ecs/Rel_A

Appendix D. Example Target MCFs

Appendix D. Example Target MCFs

This appendix contains sample Target MCF for various types of granules to be Inserted to the IMF Data Server. A Target MCF is necessary for performing any Insert.

D.1 Target MCF For Dynamic Data Granule

Listing D.1-1 contains a sample Target MCF used for Inserting a dynamic data file to the IMF Data Server.

Listing D.1-1 - Target MCF For Dynamic Data Granule

```
GROUP                   = INVENTORYMETADATA
  GROUPTYPE             = MASTERGROUP
  OBJECT                  = SHORTNAME
    VALUE                 = "MOD01"
    NUM_VAL               = 1
  END_OBJECT              = SHORTNAME
  OBJECT                  = VERSIONID
    VALUE                 = "1"
    NUM_VAL               = 1
  END_OBJECT              = VERSIONID
  OBJECT                  = SIZEMBECSDATAGRANULE
    NUM_VAL               = 1
    VALUE                 = 214.500000
  END_OBJECT              = SIZEMBECSDATAGRANULE
  GROUP                   = BOUNDINGRECTANGLE
    OBJECT                  = EASTBOUNDINGCOORDINATE
      NUM_VAL               = 1
      VALUE                 = -64.435704
    END_OBJECT              = EASTBOUNDINGCOORDINATE
    OBJECT                  = WESTBOUNDINGCOORDINATE
      NUM_VAL               = 1
      VALUE                 = -94.753015
    END_OBJECT              = WESTBOUNDINGCOORDINATE
    OBJECT                  = NORTHBOUNDINGCOORDINATE
      NUM_VAL               = 1
      VALUE                 = 44.189227
    END_OBJECT              = NORTHBOUNDINGCOORDINATE
    OBJECT                  = SOUTHBOUNDINGCOORDINATE
      NUM_VAL               = 1
```

Listing D.1-1 - Target MCF For Dynamic Data Granule (cont.)

```
      VALUE                = 23.161212
    END_OBJECT             = SOUTHBOUNDINGCOORDINATE
  END_GROUP              = BOUNDINGRECTANGLE
  GROUP                  = GPOLYGON
    OBJECT                 = GRINGCONTAINER
      OBJECT                 = EXCLUSIONGRINGFLAG
        NUM_VAL              = 1
        VALUE                = "N"
      END_OBJECT             = EXCLUSIONGRINGFLAG
      OBJECT                 = GRINGPOINTLATITUDE
        NUM_VAL              = 4
        VALUE                = (44.189227, 40.126979, 23.161212, 26.484776)
      END_OBJECT             = GRINGPOINTLATITUDE
      OBJECT                 = GRINGPOINTLONGITUDE
        NUM_VAL              = 4
        VALUE                      = (-92.656359, -64.435704, -71.747954, -
94.753015)
      END_OBJECT             = GRINGPOINTLONGITUDE
      OBJECT                 = GRINGPOINTSEQUENCENO
        NUM_VAL              = 4
        VALUE                = ("1", "2", "3", "4")
      END_OBJECT             = GRINGPOINTSEQUENCENO
    END_OBJECT             = GRINGCONTAINER
  END_GROUP              = GPOLYGON
  GROUP                  = ORBITCALCULATEDSPATIALDOMAIN
    OBJECT                 = ORBITNUMBER
      NUM_VAL              = 1
      VALUE                = 1
    END_OBJECT             = ORBITNUMBER
```

Listing D.1-1 - Target MCF For Dynamic Data Granule (cont.)

```
END_GROUP              = ORBITCALCULATEDSPATIALDOMAIN
GROUP                  = RANGEDATETIME
  OBJECT                 = RANGEBEGINNINGDATE
    NUM_VAL              = 1
    VALUE                = "1996-07-30"
  END_OBJECT             = RANGEBEGINNINGDATE
  OBJECT                 = RANGEBEGINNINGTIME
    NUM_VAL              = 1
    VALUE                = "00:00:00Z"
  END_OBJECT             = RANGEBEGINNINGTIME
  OBJECT                 = RANGEENDINGDATE
    NUM_VAL              = 1
    VALUE                = "1996-07-30"
  END_OBJECT             = RANGEENDINGDATE
  OBJECT                 = RANGEENDINGTIME
    NUM_VAL              = 1
    VALUE                = "01:00:00Z"
  END_OBJECT             = RANGEENDINGTIME
END_GROUP              = RANGEDATETIME
GROUP                  = QASTATS
  OBJECT                 = QAPERCENTINTERPOLATEDDATA
    NUM_VAL              = 1
    VALUE                = 0
  END_OBJECT             = QAPERCENTINTERPOLATEDDATA
  OBJECT                 = QAPERCENTOUTOFBOUNDSDATA
    NUM_VAL              = 1
    VALUE                = 0
  END_OBJECT             = QAPERCENTOUTOFBOUNDSDATA
  OBJECT                 = QAPERCENTMISSINGDATA
```

Listing D.1-1 - Target MCF For Dynamic Data Granule (cont.)

```
    NUM_VAL               = 1
    VALUE                 = 0
  END_OBJECT              = QAPERCENTMISSINGDATA
END_GROUP               = QASTATS
GROUP                   = QACOLLECTIONSTATS
  OBJECT                  = AUTOMATICQUALITYFLAG
    NUM_VAL               = 1
    VALUE                 = "passed"
  END_OBJECT              = AUTOMATICQUALITYFLAG
  OBJECT                  = OPERATIONALQUALITYFLAG
    NUM_VAL               = 1
    VALUE                 = "not being investigated"
  END_OBJECT              = OPERATIONALQUALITYFLAG
  OBJECT                  = SCIENCEQUALITYFLAG
    NUM_VAL               = 1
    VALUE                 = "not being investigated"
  END_OBJECT              = SCIENCEQUALITYFLAG
  OBJECT                  = QUALITYFLAGEXPLANATION
    NUM_VAL               = 1
    VALUE                 = "not being investigated"
  END_OBJECT              = QUALITYFLAGEXPLANATION
END_GROUP               = QACOLLECTIONSTATS
GROUP                   = ECSDATAGRANULE
  OBJECT                  = REPROCESSINGACTUAL
    NUM_VAL               = 1
    VALUE                 = "processed once"
  END_OBJECT              = REPROCESSINGACTUAL
  OBJECT                  = REPROCESSINGPLANNED
    NUM_VAL               = 1
```

Listing D.1-1 - Target MCF For Dynamic Data Granule (cont.)

```
      VALUE                = "no further update anticipated"
    END_OBJECT             = REPROCESSINGPLANNED
  END_GROUP              = ECSDATAGRANULE
  GROUP                  = INPUTGRANULE
    OBJECT                 = INPUTPOINTER
      NUM_VAL              = 7
      VALUE                   = ("L1A_Leading_0", "L1A__1", "L1A_Trailing_2",
"sd.coeff.trend",  "L1B.control.params",  "Reflective_Lookup_Tables_f",
"Emissive_Lookup_Tables_f")
    END_OBJECT             = INPUTPOINTER
  END_GROUP              = INPUTGRANULE
  OBJECT                 = OPERATIONMODE
    NUM_VAL              = 1
    VALUE                = "day"
  END_OBJECT             = OPERATIONMODE
END_GROUP              = INVENTORYMETADATA
END
```

D.2 Target MCF For Static Data Granule

Listing D.2-1 contains a sample Target MCF used for Inserting a static data file to the IMF Data Server. Note that the metadata attribute LocalGranuleID is used for storing the file names of all the static data granules that will be Inserted to the MOD02LUT collection (this same Target MCF can be used for each). This is optional, but can be useful.

Listing D.2-1 - Target MCF For Static Data Granule

```
GROUP= INVENTORYMETADATA


GROUPTYPE= MASTERGROUP


OBJECT= ShortName
  VALUE= "MOD02LUT"
  NUM_VAL= 1
END_OBJECT= ShortName


OBJECT= VersionID
  VALUE= "1"
  NUM_VAL= 1
END_OBJECT= VersionID



GROUP= INFORMATIONCONTENT
  OBJECT = INFORMATIONCONTENTCONTAINER


    CLASS = "1"


    OBJECT= PARAMETERNAME
      NUM_VAL = 1
      CLASS = "1"
      VALUE = "LocalGranuleID"
    END_OBJECT= PARAMETERNAME


    OBJECT= PARAMETERVALUE
```

Listing D.2-1 - Target MCF For Static Data Granule (cont.)

```
      NUM_VAL = 10

      CLASS = "1"

VALUE  =  ("Reflective_Lookup_Tables_f",  "Emissive_Lookup_Tables_f",
"L1B.params", "L1B.control.params")

    END_OBJECT= PARAMETERVALUE


  END_OBJECT = INFORMATIONCONTENTCONTAINER

END_GROUP= INFORMATIONCONTENT



END_GROUP=INVENTORYMETADATA

END
```

D.3 Target MCF For Static MCF Granule

Listing D.3-1 contains a sample Target MCF used for Inserting a MCF file to the IMF Data Server. Note that the metadata attribute LocalGranuleID is used for storing the file name of the data granule (a MCF in this case). This is optional, but can be useful.

The MCF can be optionally Inserted to the same collection (ESDT) as is defined for static data granules. In other words, one “bucket” ESDT may be defined for MCFs as well as static input data granules. This can be done since the Target MCFs are essentially the same.

Listing D.3-1 - Target MCF for Static MCF Granule

```
GROUP                  = INVENTORYMETADATA
  GROUPTYPE            = MASTERGROUP


  OBJECT                 = SHORTNAME
    NUM_VAL              = 1
    VALUE                = "MODMCF"
  END_OBJECT             = SHORTNAME


  OBJECT                 = VERSIONID
    NUM_VAL              = 1
    VALUE                = "1"
  END_OBJECT             = VERSIONID



  GROUP                  = INFORMATIONCONTENT


    OBJECT                 = INFORMATIONCONTENTCONTAINER
      CLASS                = "1"


      OBJECT                 = PARAMETERNAME
        CLASS                = "1"
        NUM_VAL              = 1
        VALUE                = "LocalGranuleID"
      END_OBJECT             = PARAMETERNAME


      OBJECT                 = PARAMETERVALUE
        CLASS                = "1"
```

Listing D.3-1 - Target MCF for Static MCF Granule (cont.)

```
        NUM_VAL               = 1
        VALUE                 = "MOD.AM1.V1.seaice_max.L2.v.mcf"
      END_OBJECT              = PARAMETERVALUE

      OBJECT                  = PARAMETERNAME
        CLASS                 = "2"
        NUM_VAL               = 1
        VALUE                 = "Science_Group"
      END_OBJECT              = PARAMETERNAME

      OBJECT                  = PARAMETERVALUE
        CLASS                 = "2"
        NUM_VAL               = 1
        VALUE                 = "M1"
      END_OBJECT              = PARAMETERVALUE

    END_OBJECT              = INFORMATIONCONTENTCONTAINER

  END_GROUP              = INFORMATIONCONTENT

END_GROUP              = INVENTORYMETADATA

END
```

D.4 Target MCF For Science Software Executable Package (SSEP)

Listing D.4-1 contains a sample Target MCF used for Inserting the SSEP (Science Software Executable Package, the PGE executable tar file) to the IMF Data Server.

A separate ESDT can be defined for all PGEs from a particular instrument.

Listing D.4-1 - Target MCF for Science Software Executable Package (SSEP)

```
GROUP                   = INVENTORYMETADATA
  GROUPTYPE             = MASTERGROUP

  OBJECT                  = SHORTNAME
    NUM_VAL               = 1
    VALUE                 = "PGEEXE"
  END_OBJECT              = SHORTNAME

  OBJECT                  = VERSIONID
    NUM_VAL               = 1
    VALUE                 = "1"
  END_OBJECT              = VERSIONID


  GROUP                   = INFORMATIONCONTENT

    OBJECT                  = INFORMATIONCONTENTCONTAINER
      CLASS                 = "1"

      OBJECT                  = PARAMETERNAME
        CLASS                 = "1"
        NUM_VAL               = 1
        VALUE                 = "ExePGEName"
      END_OBJECT              = PARAMETERNAME

      OBJECT                  = PARAMETERVALUE
        CLASS                 = "1"
```

Listing D.4-1 - Target MCF for Science Software Executable Package (SSEP) (cont.)

```
  NUM_VAL                  = 1
  VALUE                    = "PGE08"
END_OBJECT                 = PARAMETERVALUE

OBJECT                     = PARAMETERNAME
  CLASS                    = "2"
  NUM_VAL                  = 1
  VALUE                    = "ExePGESSWVersion"
END_OBJECT                 = PARAMETERNAME

OBJECT                     = PARAMETERVALUE
  CLASS                    = "2"
  NUM_VAL                  = 1
  VALUE                    = "1"
END_OBJECT                 = PARAMETERVALUE

OBJECT                     = PARAMETERNAME
  CLASS                    = "3"
  NUM_VAL                  = 1
  VALUE                    = "ExePlatformOS"
END_OBJECT                 = PARAMETERNAME

OBJECT                     = PARAMETERVALUE
  CLASS                    = "3"
  NUM_VAL                  = 1
  VALUE                    = "IRIX"
END_OBJECT                 = PARAMETERVALUE
```

Listing D.4-1 - Target MCF for Science Software Executable Package (SSEP) (cont.)

```
OBJECT                  = PARAMETERNAME
  CLASS                 = "4"
  NUM_VAL               = 1
  VALUE                 = "ExePlatformOSVersion"
END_OBJECT              = PARAMETERNAME

OBJECT                  = PARAMETERVALUE
  CLASS                 = "4"
  NUM_VAL               = 1
  VALUE                 = "6.2"
END_OBJECT              = PARAMETERVALUE

OBJECT                  = PARAMETERNAME
  CLASS                 = "5"
  NUM_VAL               = 1
  VALUE                 = "ExeInsertDate"
END_OBJECT              = PARAMETERNAME

OBJECT                  = PARAMETERVALUE
  CLASS                 = "5"
  NUM_VAL               = 1
  VALUE                 = "970516"
END_OBJECT              = PARAMETERVALUE

OBJECT                  = PARAMETERNAME
  CLASS                 = "6"
  NUM_VAL               = 1
```

Listing D.4-1 - Target MCF for Science Software Executable Package (SSEP) (cont.)

```
        VALUE                   = "ExeInsertTime"
END_OBJECT              = PARAMETERNAME

      OBJECT                    = PARAMETERVALUE
        CLASS                   = "6"
        NUM_VAL                 = 1
        VALUE                   = "15:35:00"
      END_OBJECT                = PARAMETERVALUE

    END_OBJECT                = INFORMATIONCONTENTCONTAINER

  END_GROUP                 = INFORMATIONCONTENT

END_GROUP               = INVENTORYMETADATA

END
```

Appendix E. Example PDPS ODL Files

Appendix E. Example PDPS ODL Files

This appendix contains example PDPS ODL files that are used in the PGE registration process. Detail procedures for registering a PGE are in Section 11.

E.1 PGE ODL File and PCF

Listing E.1-1 is a Process Control File for a delivered PGE. It has been updated to reflect testing in the DAAC environment. After running the DpAtCreateOdlTemplate.sh tool, the output ODL file is hand edited. The resulting ODL file is shown in Listing E.1-2. See Section 11.1.2 for details.

Listing E.1-1- Process Control File for a PGE

```
#      PGE08.pcf
#
# description:
#      Process Control File (PCF)
#
# notes:
#
#      This file supports the IR-1 version of the toolkit.
#
#      Please treat this file as a master template and make copies of it
#      for your own testing. Note that the Toolkit installation script sets
#      PGS_PC_INFO_FILE to point to this master file by default. Remember
#      to reset the environment variable PGS_PC_INFO_FILE to point to the
#      instance of your PCF.
#
# -----------------------------------------------------------------------
?   SYSTEM RUNTIME PARAMETERS
# -----------------------------------------------------------------------
# Production Run ID - unique production instance identifier
# -----------------------------------------------------------------------
1
# -----------------------------------------------------------------------
# Software ID - unique software configuration identifier
# -----------------------------------------------------------------------
1
#
?   PRODUCT INPUT FILES
# Next non-comment line is the default location for PRODUCT INPUT FILES
# WARNING! DO NOT MODIFY THIS LINE unless you have relocated these
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
# data set files to the location specified by the new setting.

!  ~/runtime

# Product Input Files (HDF format)

700000|MOD.AM1.V1.radiance.L1B.D1996212.162027.D1996257|/vola0/egs_copy/MODIS/RUN/inpu
t/simdata/L1B/32_day/day01||||1

600000|MOD.AM1.V1.geoloc.L1A.D1996212.162027.D1996257|/vola0/egs_copy/MODIS/RUN/input/
simdata/L1B/32_day/day01|||MOD.AM1.V1.geoloc.L1A.D1996212.162027.D1996257|1

#

# -----------------------------------------------------------------------

# These are actual ancillary data set files - supplied by ECS or the user

# the following are supplied for purposes of tests and as a useful set of

# ancillary data.

# -----------------------------------------------------------------------

10780|usatile12||||10751|12

10780|usatile11||||10750|11

10780|usatile10||||10749|10

10780|usatile9||||10748|9

10780|usatile8||||10747|8

10780|usatile7||||10746|7

10780|usatile6||||10745|6

10780|usatile5||||10744|5

10780|usatile4||||10743|4

10780|usatile3||||10742|3

10780|usatile2||||10741|2

10780|usatile1||||10740|1

10951|mowe13a.img|||||1

10952|owe13a.img|||||1

10953|owe14d.img|||||1

10954|owe14dr.img|||||1
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
10955|etop05.dat|||||1

10956|fnocazm.img|||||1

10957|fnococm.img|||||1

10958|fnocpt.img|||||1

10959|fnocrdg.img|||||1

10960|fnocst.img|||||1

10961|fnocurb.img|||||1

10962|fnocwat.img|||||1

10963|fnocmax.imgs|||||1

10964|fnocmin.imgs|||||1

10965|fnocmod.imgs|||||1

10966|srzarea.img|||||1

10967|srzcode.img|||||1

10968|srzphas.img|||||1

10969|srzslop.img|||||1

10970|srzsoil.img|||||1

10971|srztext.img|||||1

10972|nmcRucPotPres.datrepack|||||1

10973|tbase.bin||||10915|1

10974|tbase.br||||10919|4

10974|tbase.bl||||10918|3

10974|tbase.tr||||10917|2

10974|tbase.tl||||10916|1

10975|geoid.dat|||||1

#

# -----------------------------------------------------------------------

# The following are for the PGS_GCT tool only.

# The IDs are #defined in the PGS_GCT.h file

# -----------------------------------------------------------------------
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
10200|nad27sp|~/runtime||||1

10201|nad83sp|~/runtime||||1

# -----------------------------------------------------------------------

# The following are for the PGS_AA_DCW tool only.

# The IDs are #defined in the PGS_AA_DCW.h file

# -----------------------------------------------------------------------

10990|eurnasia/|||||1

10991|noamer/|||||1

10992|soamafr/|||||1

10993|sasaus/|||||1

#

# -----------------------------------------------------------------------

# file for Constant & Unit Conversion (CUC) tools

# IMPORTANT NOTE: THIS FILE WILL BE SUPPLIED AFTER TK4 DELIVERY!

# -----------------------------------------------------------------------

10999|PGS_CUC_maths_parameters|~/runtime||||1

#

#

#-------------------------------------------------------------------

# Metadata Configuration File (MCF) is a template to be filled in by the

# Instrument teams.

#-------------------------------------------------------------------

10250|MCF|||||1

10252|GetAttr.tmp|/vola0/egs_copy/MODIS/RUN/output/PGE08/Function/1/tmp||||1

10254|MCFWrite.tmp|/vola0/egs_copy/MODIS/RUN/output/PGE08/Function/1/tmp||||1

# MCF file

229500|MOD.AM1.V1.seaice_max.L2.mcf|/vobhome/SDST/MODIS/STORE/PGE08/MOD_PR29/source|||
|1

#
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
#
# -----------------------------------------------------------------------
# Ephemeris and Attitude files logical IDs
# Emphemeris files will be accessed via the logical ID 10501.
# Attitude files will be accessed via the logical ID 10502.
# Use file versions to allow for multiple physical ephemeris
# or attitude files.
# -----------------------------------------------------------------------
#
10501|INSERT_EPHEMERIS_FILES_HERE|||||1
10502|INSERT_ATTITUDE_FILES_HERE|||||1
#
?   PRODUCT OUTPUT FILES
# Next line is the default location for PRODUCT OUTPUT FILES
! /vola0/egs_copy/MODIS/RUN/output/PGE08/Function/1/run_time
# Product Output Files (HDF format)
229100|MOD.AM1.V1.seaice_max.L2.D1996212.162027|/vola0/egs_copy/MODIS/RUN/output/PGE08
/Function/1/run_time||||1
#
#-------------------------------------------------------------------
# This file is created when PGS_MET_Write is used with an intention
# to write an ASCII representation of the MCF in memory. The user is
# allowed to change the name and path if required
#
#-------------------------------------------------------------------
10255|asciidump|||||1
# ----------------------------------------------------------------------
#
?   SUPPORT INPUT FILES
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
# Next line is the default location for SUPPORT INPUT FILES

!  ~/runtime

#

#

# -----------------------------------------------------------------------

# This ID is #defined in PGS_AA_Tools.h

# This file contains the IDs for all support and format files shown above

# -----------------------------------------------------------------------

10900|indexFile|~/runtime||||1

#

# -----------------------------------------------------------------------

# These are support files for the data set files - to be created by user

# (not necessar

# The IDs must correspond to the logical IDs in the index file

# -----------------------------------------------------------------------

10901|mowe13aSupport|~/runtime||||1

10902|owe13aSupport|~/runtime||||1

10903|owe14Support|~/runtime||||1

10904|etop05Support|~/runtime||||1

10905|fnoc1Support|~/runtime||||1

10906|fnoc2Support|~/runtime||||1

10907|zobler1Support|~/runtime||||1

10908|zobler2Support|~/runtime||||1

10909|nmcRucSupport|~/runtime||||1

10915|tbaseSupport|~/runtime||||1

10916|tbase1Support|~/runtime||||1

10917|tbase2Support|~/runtime||||1

10918|tbase3Support|~/runtime||||1

10919|tbase4Support|~/runtime||||1
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
10740|usatile1Support|~/runtime||||1
10741|usatile2Support|~/runtime||||1
10742|usatile3Support|~/runtime||||1
10743|usatile4Support|~/runtime||||1
10744|usatile5Support|~/runtime||||1
10745|usatile6Support|~/runtime||||1
10746|usatile7Support|~/runtime||||1
10747|usatile8Support|~/runtime||||1
10748|usatile9Support|~/runtime||||1
10749|usatile10Support|~/runtime||||1
10750|usatile11Support|~/runtime||||1
10751|usatile12Support|~/runtime||||1
10948|geoidSupport|~/runtime||||1
#
# ----------------------------------------------------------------------
# The following are format files for each data set file
# (not necessarily a one-to-one relationship)
# The IDs must correspond to the logical IDs in the index file
# ----------------------------------------------------------------------
10920|mowe13a.bfm|~/runtime||||1
10921|owe13a.bfm|~/runtime||||1
10922|owe14d.bfm|~/runtime||||1
10923|owe14dr.bfm|~/runtime||||1
10924|etop05.bfm|~/runtime||||1
10925|fnocAzm.bfm|~/runtime||||1
10926|fnocOcm.bfm|~/runtime||||1
10927|fnocPt.bfm|~/runtime||||1
10928|fnocRdg.bfm|~/runtime||||1
10929|fnocSt.bfm|~/runtime||||1
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
10930|fnocUrb.bfm|~/runtime||||1

10931|fnocWat.bfm|~/runtime||||1

10932|fnocMax.bfm|~/runtime||||1

10933|fnocMin.bfm|~/runtime||||1

10934|fnocMod.bfm|~/runtime||||1

10935|srzArea.bfm|~/runtime||||1

10936|srzCode.bfm|~/runtime||||1

10937|srzPhas.bfm|~/runtime||||1

10938|srzSlop.bfm|~/runtime||||1

10939|srzSoil.bfm|~/runtime||||1

10940|srzText.bfm|~/runtime||||1

10941|nmcRucSigPotPres.bfm|~/runtime||||1

10942|tbase.bfm|~/runtime||||1

10943|tbase1.bfm|~/runtime||||1

10944|tbase2.bfm|~/runtime||||1

10945|tbase3.bfm|~/runtime||||1

10946|tbase4.bfm|~/runtime||||1

10700|usatile1.bfm|~/runtime||||1

10701|usatile2.bfm|~/runtime||||1

10702|usatile3.bfm|~/runtime||||1

10703|usatile4.bfm|~/runtime||||1

10704|usatile5.bfm|~/runtime||||1

10705|usatile6.bfm|~/runtime||||1

10706|usatile7.bfm|~/runtime||||1

10707|usatile8.bfm|~/runtime||||1

10708|usatile9.bfm|~/runtime||||1

10709|usatile10.bfm|~/runtime||||1

10710|usatile11.bfm|~/runtime||||1

10711|usatile12.bfm|~/runtime||||1
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
10947|geoidbfm|~/runtime||||1

#

#

# -----------------------------------------------------------------------

# leap seconds (TAI-UTC) file

# -----------------------------------------------------------------------

10301|leapsec.dat|~/database/sgi32/TD||||1

#

# -----------------------------------------------------------------------

# polar motion and UTC-UT1 file

# -----------------------------------------------------------------------

10401|utcpole.dat|~/database/sgi32/CSC||||1

#

# -----------------------------------------------------------------------

# earth model tags file

# -----------------------------------------------------------------------

10402|earthfigure.dat|~/database/sgi32/CSC||||1

#

# -----------------------------------------------------------------------

# JPL planetary ephemeris file (binary form)

# -----------------------------------------------------------------------

10601|de200.eos|~/database/sgi32/CBP||||1

#

#

?   SUPPORT OUTPUT FILES

# Next line is default location for SUPPORT OUTPUT FILES

! /vola0/egs_copy/MODIS/RUN/output/PGE08/run_logs

#

#
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
# -----------------------------------------------------------------------
# These files support the SMF log functionality. Each run will cause
# status information to be written to 1 or more of the Log files. To
# simulate DAAC operations, remove the 3 Logfiles between test runs.
# Remember: all executables within a PGE will contribute status data to
# the same batch of log files.
# -----------------------------------------------------------------------
10100|LogStatus|||||1
10101|LogReport|||||1
10102|LogUser|||||1
10103|TmpStatus|||||1
10104|TmpReport|||||1
10105|TmpUser|||||1
10110|MailFile|||||1
#
# -----------------------------------------------------------------------
# This paramater controls the Event Logger connection from the Toolkit.
# -----------------------------------------------------------------------
10113|eventLogger.log|||||1
#
# -----------------------------------------------------------------------
# ASCII file which stores pointers to runtime SMF files in lieu of
# loading them to shared memory, which is a TK5 enhancement.
# -----------------------------------------------------------------------
10111|ShmMem|||||1
#
#
?   USER DEFINED RUNTIME PARAMETERS
#
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
#
# -----------------------------------------------------------------------
# These parameters are required to support the PGS_SMF_Send...() tools.
# If the first parameter (TransmitFlag) is disabled, then none of the
# other parameters need to be set. By default, this functionality has been
# disabled. To enable, set TransmitFlag to 1 and supply the other 3
# parameters with local information.
# -----------------------------------------------------------------------
10109|TransmitFlag; 1=transmit,0=disable|0
10106|RemoteHost|sandcrab
10107|RemotePath|/usr/kwan/test/PC/data
10108|EmailAddresses|kwan@@eos.hitc.com
#
# -----------------------------------------------------------------------
# This paramater controls the Event Logger connection from the Toolkit.
# -----------------------------------------------------------------------
10112|Event Logging Flag; 1=connect,0=disconnect|1
#
#
# -----------------------------------------------------------------------
# The following runtime parameters define various logging options.
# Parameters described as lists should be space (i.e. ' ') seperated.
# -----------------------------------------------------------------------
10114|Logging Control; 0=disable logging, 1=enable logging|1
10115|Trace Control; 0=no trace, 1=error trace, 2=full trace|1
10116|Process ID logging; 0=don't log PID, 1=log PID|1
10117|Disabled status level list (e.g. W S F)|W
10118|Disabled seed list|1
10119|Disabled status code mnemonic list|1
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
#
# -----------------------------------------------------------------------
# The following parameters define the ADEOS-II TMDF values (all values
# are assumed to be floating point types).  The ground reference time
# should be in TAI93 format (SI seconds since 12 AM UTC 1993-01-01).
# These formats are only prototypes and are subject to change when
# the ADEOS-II TMDF values are clearly defined.
# -----------------------------------------------------------------------
10120|ADEOS-II s/c reference time|1
10121|ADEOS-II ground reference time|1
10122|ADEOS-II s/c clock period|1
#
# -----------------------------------------------------------------------
# The following parameter defines the TRMM UTCF value (the value is
# assumed to be a floating point type).
# -----------------------------------------------------------------------
10123|TRMM UTCF value|1
#
# -----------------------------------------------------------------------
# The following parameter defines the Epoch date to be used for the
# interpretation (conversion) of NASA PB5C times (the Epoch date should
# be specified here in CCSDS ASCII format--A or B)
# -----------------------------------------------------------------------
10124|NASA PB5C time Epoch date (ASCII UTC)|1
#
# -----------------------------------------------------------------------
# This entry defines the IP address of the processing host and is used
# by the Toolkit when generating unique Intermediate and Temporary file
# names.  The Toolkit no longer relies on the PGS_HOST_PATH environment
```

Listing E.1-1- Process Control File for a PGE (cont.)

```
# variable to otain this information.

# -----------------------------------------------------------------------

10099|Local IP Address of 'ether'|155.157.31.87

#

?   INTERMEDIATE INPUT

# Next line is default location for INTERMEDIATE INPUT FILES

!  ~/runtime

#

#

?   INTERMEDIATE OUTPUT

# Next line is default location for INTERMEDIATE OUTPUT FILES

! /vola0/egs_copy/MODIS/RUN/output/PGE08/Function/1/tmp

#

#

?   TEMPORARY I/O

# Next line is default location for TEMPORARY FILES

! /vola0/egs_copy/MODIS/RUN/output/PGE08/Function/1/tmp

#

#

?   END
```

Listing E.1-2 - Example Corresponding PGE ODL File

```
PGE_NAME = "PGE08"

PGE_VERSION = "1"

PGE_SSW_VERSION = "1"

PLATFORM = "EOSAM1"

INSTRUMENT = "MODIS"

PROCESSING_PERIOD = "HOURS=1"

PROCESSING_BOUNDARY = "START_OF_HOUR"

OBJECT = PCF_ENTRY

   CLASS = 1

   LOGICAL_ID = 700000

   PCF_FILE_TYPE = 1

   DATA_TYPE_NAME = "MOD02"

   DATA_TYPE_REQUIREMENT = 1

END_OBJECT = PCF_ENTRY

OBJECT = PCF_ENTRY

   CLASS = 2

   LOGICAL_ID = 600000

   PCF_FILE_TYPE = 1

   DATA_TYPE_NAME = "MOD03"

   DATA_TYPE_REQUIREMENT = 1

END_OBJECT = PCF_ENTRY

OBJECT = PCF_ENTRY

   CLASS = 3

   LOGICAL_ID = 229500

   PCF_FILE_TYPE = 1

   DATA_TYPE_NAME = "MODMCF"

   DATA_TYPE_REQUIREMENT = 1

   SCIENCE_GROUP = "M1"

END_OBJECT = PCF_ENTRY
```

Listing E.1-2 - Example Corresponding PGE ODL File (cont.)

```
OBJECT = PCF_ENTRY

   CLASS = 4

   LOGICAL_ID = 229100

   PCF_FILE_TYPE = 2

   DATA_TYPE_NAME = "MOD29"

   YIELD = 1

   SCIENCE_GROUP = "S1"

END_OBJECT = PCF_ENTRY

END
```

E.2 ESDT ODL File For Dynamic Data Granule

The listing below is an example ESDT ODL file for a dynamic data file whose temporal coverage is 24 hours.

Refer to Section 11.1.1 for detailed procedures on constructing an ESDT ODL file for PDPS.

Listing E.2-1 - Example ESDT ODL File for a Dynamic Data Granule

```
/**********************************************************************/
/*                                                                    */
/*              TEMPLATE ESDT SCIENCE METADATA ODL FILE               */
/*                                                                    */
/*                                                                    */
/* The SSIT operator's responsibility is to copy this file over and   */
/* edit it to add all necessary PDPS metadata values.                 */
/*                                                                    */
/* Each ESDT used by a PGE must have a corresponding ESDT SCIENCE     */
/* metadata ODL file.                                                 */
/*                                                                    */
/* All ESDT ODL files must reside in directory $DPAT_ESDT_SCIENCE_MD. */
/*                                                                    */
/* The operator must add a value to the left of the "=" for each      */
/* parameter.                                                         */
/*                                                                    */
/*                                                                    */
/*                                                                    */
/**********************************************************************/

/**********************************************************************/
/*          Data Type name                                            */
/*             -- Must be a string, max len 8 characters              */
/*             -- ESDT name inside ODL file must be identical to      */
/*                   ESDT name used as part of ODL filename,          */
/*                   which in turn was generated from the             */
/*                   DATA_TYPE_NAME in the PGE ODL file for the PCF   */
/*                   entry                                            */
/*         Example                                                    */
```

Listing E.2-1 - Example ESDT ODL File for a Dynamic Data Granule (cont.)

```
/*            DATA_TYPE_NAME  = "CER00"                              */
/*********************************************************************/

DATA_TYPE_NAME = "zin"

/*********************************************************************/
/*          Science instrument name                                  */
/*             -- Must be a string, max 10 len characters            */
/*         Example                                                   */
/*            INSTRUMENT  = "NMC"                                    */
/*********************************************************************/

INSTRUMENT = "SYNPGE"

/*********************************************************************/
/*          Spacecraft platform name                                 */
/*             -- Must be a string, max len 20 characters            */
/*         Example                                                   */
/*           PLATFORM  = "NOAA9"                                     */
/*********************************************************************/

PLATFORM = "SYNPGE"

/*********************************************************************/
/*          ESDT description                                         */
/*             -- Must be a string, max len 60 characters            */
/*         Example                                                   */
/*           DATA_TYPE_DESCRIPTION  = "NMC 12-hour forecast"         */
/*********************************************************************/
```

Listing E.2-1 - Example ESDT ODL File for a Dynamic Data Granule (cont.)

```
DATA_TYPE_DESCRIPTION = "SYNPGE 24-hour dataset"

/*********************************************************************/
/*        ESDT data provider (DAAC name to which files are delivered) */
/*           -- Must be a string, max len 20 characters              */
/*       Example                                                     */
/*         PROVIDER  = "National Meteorological Center"              */
/*********************************************************************/

PROVIDER = "GSFC"

/*********************************************************************/
/*          Nominal ESDT file size in MB                             */
/*           -- Must be a floating point number (i.e., include a ".") */
/*           -- Must be greater than 0.000001                        */
/*         Example                                                   */
/*            NOMINAL_SIZE = 1.5                                     */
/*********************************************************************/

NOMINAL_SIZE = 0.08

/*********************************************************************/
/* THIS PARAMETER IS ONLY REQUIRED FOR files in the INPUT sections    */
/* of the PCF (PRODUCT, SUPPORT or INTERMEDIATE)                     */
/* (ignored for output files, which are always type "I")             */
/*                                                                   */
/*          Dynamic flag -- flags whether an ESDT is dynamic         */
/*             -- Allowed values:                                    */
```

Listing E.2-1 - Example ESDT ODL File for a Dynamic Data Granule (cont.)

```
/*                   "S" -- Static file                                    */
/*                   "I" -- Dynamic internal file                          */
/*                   "E" -- Dynamic external file                          */
/*         Example                                                         */
/*           DYNAMIC_FLAG  = "E"                                           */
/***********************************************************************/

DYNAMIC_FLAG = "E"

/***********************************************************************/
/* THIS PARAMETER IS ONLY REQUIRED FOR Dynamic External files         */
/* (DYNAMIC_FLAG = "E")                                                */
/*                                                                     */
/*          Data availability prediction method                        */
/*             -- Must be one of {"ROUTINE","OVERLAP"}                 */
/*         Example                                                     */
/*            PREDICTION_METHOD  = "ROUTINE"                           */
/***********************************************************************/

PREDICTION_METHOD = "ROUTINE"

/***********************************************************************/
/* THIS PARAMETER IS ONLY REQUIRED FOR Dynamic External files         */
/* (DYNAMIC_FLAG = "E")                                                */
/*                                                                     */
/*          Supplier name                                              */
/*             -- Must be a string, max len 30 characters              */
/*         Example                                                     */
/*            SUPPLIER_NAME  = "NOAA"                                  */
```

Listing E.2-1 - Example ESDT ODL File for a Dynamic Data Granule (cont.)

```
/*********************************************************************/

SUPPLIER_NAME = "GSFC"
/*********************************************************************/
/* THIS PARAMETER IS ONLY REQUIRED FOR Dynamic External files         */
/* (DYNAMIC_FLAG = "E")                                               */
/*                                                                     */
/*          Nominal collection period within granule                  */
/*          -- Must contain a single P=V string, where                */
/*                P is one of                                         */
/*                { MONTHS, WEEKS, DAYS, HOURS, MINS, SECS }          */
/*         Example                                                    */
/*            PERIOD  = "HOURS=12"                                    */
/*********************************************************************/

PERIOD = "HOURS=24"

/*********************************************************************/
/* THIS PARAMETER IS ONLY REQUIRED FOR Dynamic External files         */
/* (DYNAMIC_FLAG = "E")                                               */
/*                                                                    */
/*          Nominal time boundary on which ESDT arrives               */
/*          -- Must contain 1 or more P=V strings, where              */
/*                P is one of { START_OF_HOUR, START_OF_6HOUR,        */
/*                    START_OF_DAY, START_OF_WEEK, START_OF_MONTH };  */
/*                also, "+<n>" may be added to any of these, where <n>*/
/*                specifies integer seconds                           */
/*         Example                                                    */
/*           BOUNDARY   = "START_OF_DAY+10800"                        */
```

Listing E.2-1 - Example ESDT ODL File for a Dynamic Data Granule (cont.)

```
/*********************************************************************/


BOUNDARY = "START_OF_DAY"


/*********************************************************************/
/* THIS PARAMETER IS ONLY REQUIRED FOR Dynamic External files         */
/* (DYNAMIC_FLAG = "E")                                               */
/*                                                                    */
/*          Avg delay between granule collection and arrival, in secs */
/*             -- Must be a positive integer                          */
/*         Example                                                    */
/*            DELAY = 43200                                           */
/*********************************************************************/


DELAY = 43200


END
```

E.3 ESDT ODL File For Static Data Granule

The listing below is an example ESDT ODL file for a static data file (a lookup table).

The DATA_TYPE_NAME must be set to the ESDT ShortName of an ESDT that has been successfully registered. In this example, an ESDT with the ShortName MOD02LUT has been registered to be used for MCFs. The DYNAMIC_FLAG is set to “S” since it is a static file. Metadata checking is not required for this ESDT and that section of the template ESDT ODL file has been removed.

Refer to Section 11.1.1 for detailed procedures on constructing an ESDT ODL file for PDPS.

Listing E.3-1 - Example ESDT ODL File for a Static Data Granule

```
/**********************************************************************/
/*                                                                    */
/*              TEMPLATE ESDT SCIENCE METADATA ODL FILE               */
/*                                                                    */
/*                                                                    */
/* The SSIT operator's responsibility is to copy this file over and   */
/* edit it to add all necessary PDPS metadata values.                 */
/*                                                                    */
/* Each ESDT used by a PGE must have a corresponding ESDT SCIENCE     */
/* metadata ODL file.                                                 */
/*                                                                    */
/* All ESDT ODL files must reside in directory $DPAT_ESDT_SCIENCE_MD. */
/*                                                                    */
/* The operator must add a value to the left of the "=" for each      */
/* parameter.                                                         */
/*                                                                    */
/*                                                                    */
/*                                                                    */
/**********************************************************************/

/**********************************************************************/
/*          Data Type name                                            */
/*             -- Must be a string, max len 8 characters              */
/*             -- ESDT name inside ODL file must be identical to      */
/*                   ESDT name used as part of ODL filename,          */
/*                   which in turn was generated from the             */
/*                   DATA_TYPE_NAME in the PGE ODL file for the PCF   */
/*                   entry                                            */
/*         Example                                                    */
```

Listing E.3-1 - Example ESDT ODL File for a Static Data Granule (cont.)

```
/*            DATA_TYPE_NAME  = "CER00"                              */
/*******************************************************************/

DATA_TYPE_NAME = "MOD02LUT"

/*******************************************************************/
/*          Science instrument name                                */
/*             -- Must be a string, max 10 len characters          */
/*         Example                                                 */
/*            INSTRUMENT  = "NMC"                                  */
/*******************************************************************/

INSTRUMENT = "SYNPGE"

/*******************************************************************/
/*          Spacecraft platform name                               */
/*             -- Must be a string, max len 20 characters          */
/*         Example                                                 */
/*           PLATFORM  = "NOAA9"                                    */
/*******************************************************************/

PLATFORM = "SYNPGE"

/*******************************************************************/
/*          ESDT description                                       */
/*             -- Must be a string, max len 60 characters          */
/*         Example                                                 */
/*           DATA_TYPE_DESCRIPTION  = "NMC 12-hour forecast"       */
/*******************************************************************/
```

Listing E.3-1 - Example ESDT ODL File for a Static Data Granule (cont.)

```
DATA_TYPE_DESCRIPTION = "Lookup tables for PGE02"

/**********************************************************************/
/*        ESDT data provider (DAAC name to which files are delivered) */
/*             -- Must be a string, max len 20 characters             */
/*         Example                                                    */
/*           PROVIDER  = "National Meteorological Center"             */
/**********************************************************************/

PROVIDER = "GSFC"

/**********************************************************************/
/*          Nominal ESDT file size in MB                              */
/*           -- Must be a floating point number (i.e., include a ".") */
/*           -- Must be greater than 0.000001                         */
/*         Example                                                    */
/*            NOMINAL_SIZE = 1.5                                      */
/**********************************************************************/

NOMINAL_SIZE = 0.08

/**********************************************************************/
/* THIS PARAMETER IS ONLY REQUIRED FOR files in the INPUT sections    */
/* of the PCF (PRODUCT, SUPPORT or INTERMEDIATE)                      */
/* (ignored for output files, which are always type "I")              */
/*                                                                    */
/*          Dynamic flag -- flags whether an ESDT is dynamic          */
/*             -- Allowed values:                                     */
```

__Listing E.3-1 - Example ESDT ODL File for a Static Data Granule (cont.)__

```
/*                   "S" -- Static file                                */
/*                   "I" -- Dynamic internal file                      */
/*                   "E" -- Dynamic external file                      */
/*         Example                                                     */
/*           DYNAMIC_FLAG  = "E"                                       */
/**********************************************************************/


DYNAMIC_FLAG = "S"


END
```

E.4 ESDT ODL File For MCF

The listing below is an example ESDT ODL file for a particular type of static file, the MCF. Such a file is used by the PDPS which considers the MCF an input file for the PGE.

The DATA_TYPE_NAME must be set to the ESDT ShortName of an ESDT that has been successfully registered. In this example, an ESDT with the ShortName MODMCF has been registered to be used for MCFs. The DYNAMIC_FLAG is set to “S” since the MCF is considered a static input granule. Metadata checking is not required for this ESDT and that section of the template ESDT ODL file has been removed.

Refer to Section 11.1.1 for detailed procedures on constructing an ESDT ODL file for PDPS.

Listing E.4-1 - Example ESDT ODL File for a MCF

```
/*********************************************************************/
/*                                                                   */
/*              TEMPLATE ESDT SCIENCE METADATA ODL FILE              */
/*                                                                   */
/*                                                                   */
/* The SSIT operator's responsibility is to copy this file over and  */
/* edit it to add all necessary PDPS metadata values.                */
/*                                                                   */
/* Each ESDT used by a PGE must have a corresponding ESDT SCIENCE    */
/* metadata ODL file.                                                */
/*                                                                   */
/* All ESDT ODL files must reside in directory $DPAT_ESDT_SCIENCE_MD. */
/*                                                                   */
/* The operator must add a value to the left of the "=" for each     */
/* parameter.                                                        */
/*                                                                   */
/* NOTE: This file contains PDPS ESDT metadata.                      */
/* This metadata is NOT Data Server metadata.                        */
/*                                                                   */
/* See NOTE for ESDT PGEMISC at the end of this file.                */
/*                                                                   */
/*********************************************************************/


/*********************************************************************/
/*          Data Type name                                           */
/*             -- Must be a string, max len 8 characters             */
/*             -- Must be identical to Data Server Short Name        */
/*             -- ESDT name inside ODL file must be identical to     */
/*                   ESDT name used as part of ODL filename,         */
```

Listing E.4-1 - Example ESDT ODL File for a MCF (cont.)

```
/*                    which in turn was generated from the            */
/*                    DATA_TYPE_NAME in the PGE ODL file for the PCF  */
/*                    entry                                           */
/*          Example                                                   */
/*             DATA_TYPE_NAME  = "SHRTNAME"                           */
/*********************************************************************/

DATA_TYPE_NAME = "MODMCF"

/*********************************************************************/
/*           Science instrument name                                  */
/*              -- Must be a string, max 10 len characters            */
/*          Example                                                   */
/*             INSTRUMENT  = "CERES"                                  */
/*********************************************************************/

INSTRUMENT = "MODIS"

/*********************************************************************/
/*           Spacecraft platform name                                 */
/*              -- Must be a string, max len 20 characters            */
/*          Example                                                   */
/*            PLATFORM  = "TRMM"                                      */
/*********************************************************************/

PLATFORM = "EOSAM1"

/*********************************************************************/
/*           ESDT description                                         */
```

Listing E.4-1 - Example ESDT ODL File for a MCF (cont.)

```
/*             -- Must be a string, max len 60 characters             */
/*         Example                                                    */
/*           DATA_TYPE_DESCRIPTION  = "TRMM Level 0 data"             */
/*********************************************************************/

DATA_TYPE_DESCRIPTION = "Bucket for MCFs"

/*********************************************************************/
/*         ESDT data provider (DAAC name to which files are delivered)*/
/*             -- Must be a string, max len 100 characters            */
/*         Example                                                    */
/*           PROVIDER  = "LANGLEY"                                    */
/*********************************************************************/

PROVIDER = "GSFC"

/*********************************************************************/
/*          Nominal ESDT file size in MB                              */
/*           -- Must be a floating point number (i.e., include a ".") */
/*           -- Must be greater than 0.000001                         */
/*         Example                                                    */
/*            NOMINAL_SIZE = 1.5                                      */
/*********************************************************************/

NOMINAL_SIZE = 0.1

/*********************************************************************/
/* THIS PARAMETER IS ONLY REQUIRED FOR files in the INPUT sections    */
/* of the PCF (PRODUCT, SUPPORT or INTERMEDIATE)                      */
```

Listing E.4-1 - Example ESDT ODL File for a MCF (cont.)

```
/* (ignored for output files, which are always type "I")              */
/*                                                                    */
/*          Dynamic flag -- flags whether an ESDT is dynamic          */
/*             -- Allowed values:                                     */
/*                   "S" -- Static file                               */
/*                   "I" -- Dynamic internal file                     */
/*                   "E" -- Dynamic external file                     */
/*         Example                                                    */
/*           DYNAMIC_FLAG  = "E"                                      */
/**********************************************************************/

DYNAMIC_FLAG = "S"



END
```

Abbreviations and Acronyms

AIT	Algorithm Integration and Test
AIT	Algorithm Integration and Test (workstation)
ANSI	American National Standards Institute
API	Applications Programming Interface
ASCII	American Standard Code for Information Interchange
CM	Configuration Management
COTS	Commercial Off The Shell (hardware or software)
CSDT	Computer Science Data Type
DAAC	Distributed Active Archive Center
DAP	Delivered Algorithm Package
DBA	Database Administrator
DoD	Department of Defense
ECS	EOSDIS Core System
EOSDIS	Earth Observing System Data and Information System
ESDIS	Earth Science Data and Information System
ESDT	Earth Science Data Type
ftp	file transfer protocol
GUI	Graphical User Interface
HDF	Hierarchical Data Format
IDL	Interactive Data Language
IMF	Integrated Metastorage Facility
IRIX	Operating system name for the SGI. IRIX 6.2 is the current baselined version for the Pre-Release B Testbed.
MB	megabytes
MCF	Metadata Configuration File

NASA	National Aeronautics and Space Administration
ODL	Object Description Language
PCF	Process Control File
PDPS	Planning and Data Processing System
PGE	Product Generation Executive
PGM	Portable Grey Map (image file format)
PH	Production History
PLN	Planning (Workstation)
PSA	Product Specific Attribute
SA	System Administrator
SCF	Science Computing Facility
SDP	Science Data Processing (Toolkit)
SDPS	Science Data Processing System
SGI	Silicon Graphics, Inc.
SMF	Status Message Facility
SPR	Science Processor (Workstation)
SSI&T	Science Software Integration and Test
UR	Universal Reference
URL	Universal Resource Locator
VA	VOB Administrator
VOB	Versioned Object Base
WWW	World Wide Web

Glossary

Word or Phrase	Definition
ASCII File	A data file whose contents are encoded as ASCII characters.
Binary File	A data file whose contents are in binary form (i.e. not encoded).
Configuration Management	A developer’s nightmare.
Data Visualization	The ability to examine a data file as a raw data dump, a plot, or an image.
Delivered Algorithm Package (DAP)	An ensemble of PGE source code, control files, makefiles, documentation, and other related files delivered in a package from the SCF to the DAAC.
Earth Science Data Type (ESDT)	An ESDT defines for ECS a data collection. In the Pre-Release B Testbed, an ESDT is implemented using a descriptor file that defines all metadata (granule and collection level), services, and valids associated with that ESDT. An ESDT must exist for any data granule that is to be managed by the IMF Data Server. This includes data granules as well as PGE executables and MCFs.
HDF File	A data file whose format follows the NCSA Hierarchical Data Format standard.
HDF-EOS File	An HDF file which, at a minimum, has been created using the SDP Toolkit and containing the ECS Core metadata.
Makefile	Input file for the UNIX make facility which allows a user to specify dependencies between source (and other) files, primarily for compiling and linking programs, but also used for other activities such as building documentation and installing software.

Problem Report	
Product Generation Executive (PGE)	The smallest schedulable unit of science software. A PGE may consist of a single binary executable or a number of executables wrapped by a shell script. To the ECS system, a PGE is a black box.
Profiling	The measurement of the science software’s resource usage requirements such as memory, disk space requirements, and CPU time.
Science Software	Software developed at the SCFs to generate standard (and other) science data.
Source MCF	The metadata configuration file that is input to the PGE. Some attribute values may be set in the Source MCF. Others are set by the PGE during execution.
Standards Checking	The process of checking whether or not source code follow prescribed coding standards.
Target MCF	The fully populated metadata configuration file that is output from the PGE. All attribute values are set in the Target MCF. A Target MCF is needed to Insert any file to the IMF Data Server.
Universal Reference (UR)	A unique file name tag assigned to every granule that is Inserted to the IMF Data Server.